Ingeniero Técnico en Informática de Sistemas

Escuela Técnica Superior de Ingeniería de SISTEMAS INFORMATICOS

Universidad Politécnica de Madrid

Título: Customer Relationship Management - PeopleSoft CRM
Curso: 2014 - 2015

Autor: Daniel Ballesteros Navarro

Tutor: Miguel Angel Díaz Martínez

INDICE

1 RESUMEN 5

1.1 Resumen (español) 5

1.2 Summary (english) 5

2 INTRODUCCIÓN 7

2.1 Objetivos 7

2.2 Qué es CRM 8

2.3 PeopleSoft CRM 10

3 CONCEPTOS CLAVE DEL CRM 13

3.1 Conceptos comunes a CRM 13

3.2 Descripción de entidades y conceptos clave 13

3.2.1 Cliente 13
3.2.2 Sede 16
3.2.3 Persona de contacto 17
3.2.4 Dirección 19
3.2.5 Interacción 20
3.2.6 Caso 22
3.2.7 Cuestionario 24
3.2.8 Proyecto de Gestión 26

3.3 Caso de uso: atención de un caso tipo de cliente 27

4 ARQUITECTURA DE PEOPLESOFT 31

4.1 Conceptos básicos de PeopleSoft Internet Architecture (PIA) 31

4.2 Application Server Domains 33

4.2.1 Workstation Listeners (WSL) y Workstation Handlers (WSH) 35
4.2.2 Jolt Server Listener (JSL) y Jolt Server Handler (JSH) 36
4.2.3 Request Queues (APPQ, QCKQ, QRYQ, SAMQ) 36
4.2.4 PSAPPSRV 36
4.2.5 PSQCKSRV 37
4.2.6 PSQRYSRV 37
4.2.7 PSSAMSRV 37

4.3 Servidor batch: Process Scheduler 37

4.4 Configuraciones físicas típicas 38

4.5 Opciones para el despliegue 39

4.5.1 Conexión a través de PIA 39
4.5.2 PeopleSoft Portal 40
4.5.3 Conexión desde el entorno de desarrollo 41
4.5.4 Tecnologías para la integración 41

5 MANUAL DE USUARIO: INTORDUCCIÓN A PEOPLESOFT 43

5.1 Visión General 43

5.2 Conceptos básicos de navegación 43

5.2.1 Menú lateral 43
5.2.2 Otros elementos de navegación 46

5.2.3 Esperas en el sistema 50
5.3 Acceso al sistema 50
5.4 Desconexión del sistema 52
5.5 Funciones básicas del sistema 53
5.5.1 Realización de búsquedas 53
5.5.2 Detalle de una entidad 56
6 MANUAL DE DESARROLLO 59
6.1 PeopleTools 59
6.1.1 Integración 59
6.1.2 Informes y Analíticos 60
6.1.3 Seguridad 60
6.1.4 Gestión de ciclo de vida 60
6.1.5 Experiencia de usuario 61
6.1.6 Plataforma 61
6.1.7 Desarrollo 61
6.2 PeopleCode 61
6.2.1 Tipos de Datos 62
6.2.2 Comentarios 63
6.2.3 Sentencias 64
6.2.4 Funciones 68
6.2.5 Expresiones 72
6.2.6 Variables 77
6.2.7 Operadores 81
6.3 Introducción a Application Designer 84
6.4 Creación páginas con Application Designer 88
6.4.1 Tablas de la Base de Datos de PeopleSoft 88
6.4.2 Validación de datos 88
6.4.3 Pasos básicos para la creación de aplicaciones on-line 89
6.5 Application Engine 97
6.5.1 Introducción a Application Engine 97
6.5.2 Elementos de un programa de Application Engine 97
6.6 Acceso a base de datos 103
6.6.1 Requerimientos de acceso a base de datos 103
6.6.2 Implementación 103
6.6.3 Acceso a base de datos desde acciones SQL 103
6.6.4 Acceso a base de datos desde PeopleCode 104
6.6.5 Gestión de transacciones 104
6.6.6 Buenas prácticas para el acceso a base de datos 106
6.7 Tratamiento de ficheros 106
6.7.1 Requerimientos en el tratamiento de ficheros 106
6.7.2 Implementación 107
6.8 Gestión de errores 108
6.8.1 Requerimientos en el tratamiento de errores 108
6.8.2 Implementación 108
6.9 Rearranques 111
6.9.1 Rearrancabilidad 111
6.9.2 Requerimientos de rearrancabilidad 113
6.9.3 Implementación 113

6.10 Seguridad y accesibilidad 122
6.10.1 Implementación 122

6.11 Ejecución de programas App Engine 123
6.11.1 Modo batch: Process Scheduler 123
6.11.2 Modo manual: Línea de Comandos 124
6.11.3 Modo online: CallAppEngine 127

6.12 Descripción de programas con Application Engine 128
6.12.1 Creación de un FileLayout 130
6.12.2 Creación de un Component Interface 132
6.12.3 Descripción de programas: Exportación 137
6.12.4 Descripción de programas: Importación 146
6.12.5 Programa Re-Arrancable 155
6.12.6 Descripción de programas: copia de fichero plano 163
6.12.7 Descripción de programas: copia de tabla 165

7 CONCLUSIONES Y LÍNEAS DE CONTINUACIÓN 169

7.1 Conclusiones 169

7.2 Líneas de continuación 172

BIBLIOGRAFÍA 177

1 RESUMEN

1.1 Resumen (español)

El intercambio y comercio tanto de bienes como servicios se remonta a tiempos inmemoriales dentro de la historia de la humanidad. Desde sus inicios tempranos con el intercambio o trueque de productos en el Neolítico hasta nuestra época híper globalizada, en la que existen clientes potenciales en el otro extremo del mundo, podemos decir que se ha recorrido un largo camino.

Con el paso del tiempo y la evolución de la sociedad y la tecnología, así como la evolución empresarial, se ha visto necesario la implementación de estrategias para lograr la fidelización y satisfacción de los clientes. De esta forma entendimos que ya no valía simplemente con vender un producto a un cliente, si no que si queríamos establecer una relación continúa con el mismo, debíamos lograr su satisfacción y por tanto su fidelización.

Como forma de extender la relación más allá de una simple venta, las empresas modernas empezaron a implementar diversas estrategias. De esta forma aparecieron los primeros centros de atención al cliente, las primeras aplicaciones hechas a medida para dar soporte a los clientes y por fin los sistemas CRM tal y como los concebimos hoy día.

El presente proyecto fin de carrera da una explicación de dichos sistemas indicando cuáles son sus objetos fundamentales y cómo implementan la estrategia CRM y profundiza en uno de los sistemas CRM más utilizados: PeopleSoft CRM, dando una explicación detallada de dicho sistemas así como de los conceptos y lenguaje de programación de dicho sistema CRM.

1.2 Summary (english)

The exchange and trade of goods as well and services goes back to ancient times in the history of mankind. Since its early beginning with the bartering of products in the Neolithic to our globalized hyper era, in which there are potential customers on the other side of the world, we can say that it has come a long way.

After a certain length of time, the society and technology evolution, and also the enterprise development, has been necessary to implement strategies to achieve customer loyalty and satisfaction. We understood in this way that it no longer simply worth to sell a product to a customer, otherwise if we wanted to establish a relationship continues with the same, we should ensure their satisfaction and thus their loyalty.

As a way to extend the relationship beyond a simple sale, modern enterprises began to implement several strategies. Therefore appeared the first customer service centers, the first applications tailored to support customers and finally the CRM systems as we know it today.

This final project gives an explanation of such systems by indicating what the core objects are and how to implement the CRM strategy, deeping into one of the most widely used CRM systems: PeopleSoft CRM, and also giving a detailed explanation of this system and its programming language.

2 INTRODUCCIÓN

2.1 Objetivos

Las aplicaciones para atención al cliente conocidas como CRM (Customer Relationship Management) están ampliamente implantadas dentro del ámbito empresarial. Aunque el concepto CRM es fundamentalmente una estrategia de negocio que consiste en situar al cliente en el centro del mismo, existen en el mercado diferentes aplicaciones para implementar dicha estrategia. El objetivo del presente proyecto fin de carrera es dar una visión general de los conceptos claves de los sistemas de gestión de clientes conocidos como CRM. Particularmente se hace foco en la aplicación comercial PeopleSoft CRM, una de las aplicaciones CRM más extendidas e importantes del mercado.

Como objetivos principales a tratar en el proyecto, se desarrollarán los siguientes aspectos:

- Conceptos claves y comunes de los aplicativos de gestión de clientes: descripción de entidades principales.
- Visión detallada de la arquitectura técnica del aplicativo PeopleSoft
- Manual de usuario como introducción al aplicativo on-line explicando el uso de la navegación
- Manual de desarrollo en el que se detallarán los elementos claves de programación en PeopleCode (lenguaje de programación de PeopleSoft) con ejemplos prácticos del uso de dicho lenguaje.

Tras el desarrollo del mismo, se pretende que el lector tenga claro qué es un sistema CRM, con qué objetos claves trabaja, así como tener una visión más detallada de PeopleSoft CRM como ejemplo de CRM comercial potente y ampliamente extendido.

2.2 Qué es CRM

Las siglas CRM son el acrónimo de Customer Relationship Management (Gestión de Relaciones con el Cliente). Un factor de éxito importante en cualquier compañía se basa en la relación que tenga con sus clientes. A nivel de software, el CRM es la aplicación que implementa la estrategia CRM y soporta todos los procesos que conllevan relación con el cliente, ya sea el alta de los mismo, modificación y baja, relacionar al cliente oportunidades y ofertas, así como las reclamaciones e incidencias en las que incurra el servicio o producto contratado. La característica sin duda más relevante es el almacenamiento centralizado y único de todos los clientes de la compañía, característica sobre la que se fundamentan todos los procesos CRM y que permiten tener una relación entre la empresa y el cliente.

Existen cuatro pilares fundamentales para la gestión del CRM que son el servicio y soporte, las ventas, el marketing y el análisis:

- Servicio y Soporte: El servicio y soporte a los productos contratados son claves para la satisfacción al cliente. Es obvio que los clientes esperan siempre una respuesta adecuada y rápida a las necesidades y problemas que puedan tener de ahí que este sea uno de los pilares fundamentales del CRM. La estrategia CRM pasa por potenciar el servicio que se da al cliente para conseguir un grado máximo de satisfacción y fidelización del mismo.

- Ventas: El equipo de ventas debe ser capaz de identificar y tratar adecuadamente a clientes potenciales. Gracias a los aplicativos CRM esta labor se simplifica y se dota de herramientas eficaces para poder llevar a cabo todas las gestiones referentes a ventas.

- Marketing: La gestión de campañas, la comunicación con el cliente y la segmentación de los mismos, forma parte de toda estrategia en la que queramos que el cliente sea el centro del negocio. Este es el cometido del marketing dentro del CRM.

- Análisis: El análisis continúo de la información permitirá mejorar los procesos, el negocio y la satisfacción de los clientes. El análisis proporciona experiencia

sobre cómo se realizan las actividades y cómo éstas pueden mejorar para poder incrementar la fidelización y retención a los clientes.

Customer Relationship Management

La gestión de clientes se basa en el mantenimiento en la herramienta de CRM del maestro de clientes de la compañía. De esta forma se tiene organizado y centralizado en un único sistema toda la información referente de los clientes de cara a poder identificar, atender, fidelizar o retener a los mismos. Es imprescindible mantener una base de clientes unificada para conseguir tener toda la información integrada, de manera que cualquier canal sea capaz de acceder a los datos. Gracias a esto se conseguirá una solución de gestión con una visión del cliente de 360º. La gestión de clientes debe contemplar la tipificación y segmentación de los mismos de manera que sea posible configurar reglas de negocio de tratamiento personalizado y garantizar de esta forma una adecuada atención.

Adicionalmente, es altamente recomendable que la funcionalidad de gestión de clientes del CRM operacional se integre con el resto de sistemas, de manera que se mantenga toda la información consistente a lo largo de toda la organización. Este constituirá otro de los factores claves para la completa atención al cliente. Por ejemplo, si en nuestra organización tenemos un sistema específico de facturación sería oportuna una integración entre el facturador y el CRM de cara a poder atender los problemas en factura que tengan los clientes así como poder planificar nuevas estrategias de ventas en base a la facturación o tipo de consumos que tenga el cliente.

A modo de resumen y como conceptos claves podemos decir que el concepto CRM es una estrategia de negocio, que hay aplicaciones en el mercado que implementan dicha estrategia y que esta estrategia está basada en poner al cliente en el centro del negocio en base al servicio y soporte, las ventas, el marketing y el análisis.

2.3 PeopleSoft CRM

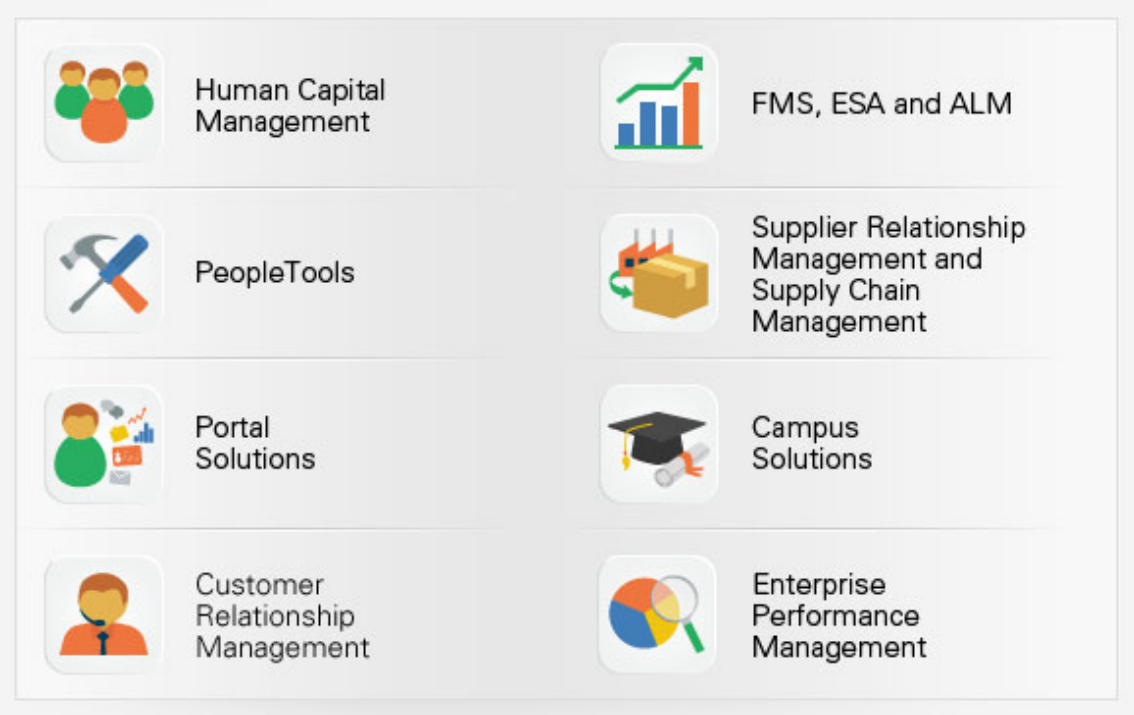
PeopleSoft CRM es una de las aplicaciones comerciales de CRM más extendidas dentro del mundo empresarial. Es una solución englobada en el producto PeopleSoft Enterprise que pertenece a la compañía Oracle Corporation desde el año 2005 tras la adquisición por parte de Oracle de la compañía americana PeopleSoft Inc. La empresa original fue creada en 1987 por Dave Duffield y Ken Morris como una empresa especializada en soluciones de Planificación de Recursos Empresariales (ERP), Recursos Humanos, Gestión de Relaciones con los Clientes (CRM) y gestión de nómina de grandes empresas. Las soluciones de PeopleSoft Inc. tuvieron un gran éxito y pronto situaron a la empresa como una de las empresas mayores y líderes en el sector. Como consecuencia y teniendo en cuenta el gran potencial, la adquisición de PeopleSoft Inc. por parte de Oracle se cerró en 10.500 millones de dólares.

Según la propia información comercial de Oracle las aplicaciones PeopleSoft Enterprise de Oracle están diseñadas para satisfacer los requisitos empresariales más complejos. Proporcionan soluciones empresariales y sectoriales completas que permiten a las organizaciones:

- Elevar la productividad
- Acelerar el rendimiento de la empresa
- Reducir el coste de propiedad

En España hay más de 400 implantaciones con soluciones PeopleSoft incluyendo implantaciones en grandes empresas y multinacionales tanto del sector financiero, del transporte así como de las telecomunicaciones.

La versión comercial actual de PeopleSoft Enterprise es la 9.2, lanzada al mercado en Marzo de 2013, y supone la cuarta versión lanzada por Oracle desde la adquisición. Esta versión se compone de los siguientes módulos:



Módulos de PeopleSoft Enterprise

Las características de estos módulos son:

- Human Capital Management (HCM): Solución focalizada en el ámbito de los recursos humanos para la mejora de procesos de negocio

- Financial Management (FMS), Enterprise Services Automation (ESA) y Asset Lifecycle Management (ALM): Solución de gestión para el ámbito financiero que soportan ciclo de vida teniendo en cuenta las operaciones, los activos o cualquier aspecto de las finanzas.
- PeopleTools (PT): Plataforma de desarrollo y ejecución de aplicaciones PeopleSoft.
- Supplier Relationship Management (SRM) y Supply Chain Management (SCM): Solución para la cadena de suministro que permite la conexión entre proveedores y clientes con los procesos de negocio.
- Portal Solutions (PS): Solución de portal para integrar los procesos de negocio de PeopleSoft.
- Campus Solutions (CS): Solución para el mundo académico de enseña superior que permite el seguimiento completo del ciclo de vida de estudiantes así como el manejo de planes educativos.
- Customer Relationship Management (CRM): Solución para la gestión de clientes en la que se ha basado el presente proyecto fin de carrera.
- Enterprise Performance Management (EPM): Solución de gestión de rendimiento para formular estrategias de crecimiento rentable, alinear estrategias con planes operativos y supervisar las operaciones del día a día.

Oracle ha desarrollado un portal específico en el que se puede obtener una amplia información sobre todos los productos PeopleSoft. Se puede consultar dicho portal en el siguiente enlace:

http://docs.oracle.com/cd/E52319_01/infoportal/index.html

En este portal se encuentra una extensa información tanto a nivel comercial como a nivel técnico con recursos, documentación y ayuda para clientes y desarrolladores. También se pueden encontrar diversos tutoriales así como las últimas noticias y eventos sobre el mundo PeopleSoft.

3 CONCEPTOS CLAVE DEL CRM

3.1 Conceptos comunes a CRM

Aunque cada aplicación CRM tiene una serie de entidades y objetos propios, bien es cierto que gran parte de estos suelen ser comunes a todas las aplicaciones CRM. De esta forma se puede realizar la atención al cliente independiente de la versión comercial que se trate. Estas entidades y conceptos básicos son:

- Cliente
- Sede
- Persona de contacto
- Dirección
- Interacción
- Caso
- Cuestionario

En este proyecto fin de carrera nos vamos a centrar en estas identidades basadas en el modelo de datos de PeopleSoft, aunque son análogas, con alguna pequeña diferencia, entre la mayoría de las aplicaciones comerciales CRM.

Además también se va a explicar, debido a su utilidad e importancia en la atención del cliente, el objeto Proyecto de Gestión de PeopleSoft CRM.

3.2 Descripción de entidades y conceptos clave

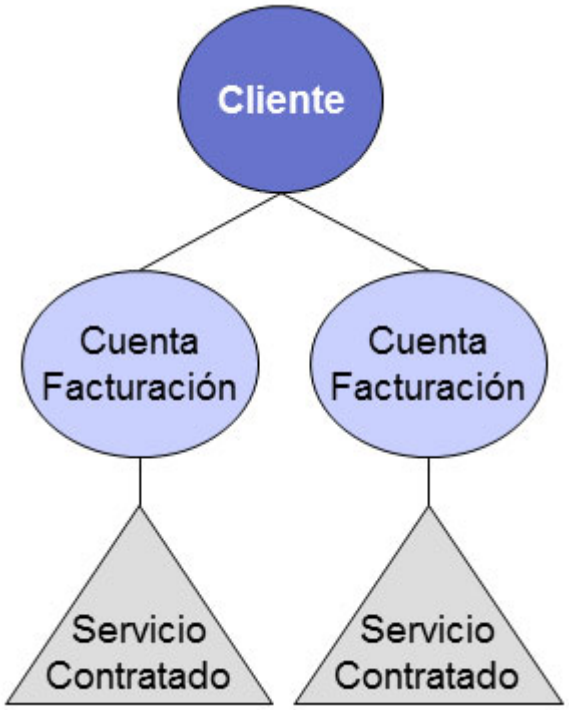
3.2.1 Cliente

El Cliente es la entidad principal del CRM. Habitualmente se define como cliente a una persona física (residencial) o jurídica (empresa) que adquiere mediante contrato o compra directa un servicio/producto.

Se pueden establecer distintas tipologías de clientes dependiendo del segmento al que pertenezcan. De esta forma se pueden tener diferenciados tanto los clientes residenciales,

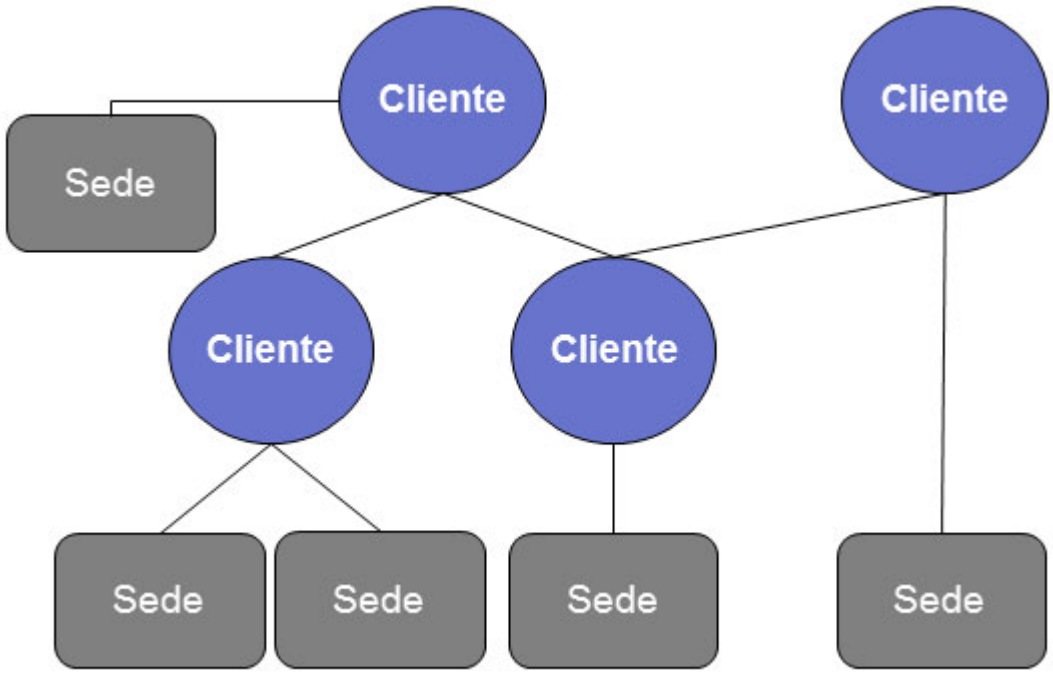
como empresas, autónomos, resellers (revendedores) u otras clasificaciones dependiendo de la segmentación deseada.

El cliente puede relacionarse y por tanto estructurarse con otras entidades del CRM. La principal jerarquía de cliente se estable con las cuentas, que son agrupaciones de productos o servicios que tiene el cliente y que pueden ser facturables o no. Una estructura habitual de cliente pudiera ser un cliente con n cuentas de facturación colgando jerárquicamente de él y cada una de estas cuentas con uno o n productos/servicios contratados.



Jerarquía de Cliente: Cliente – Cuenta – Servicio

Existe la posibilidad de establecer relaciones jerárquicas entre los propios clientes. Por ejemplo, un cliente podría revender servicios a otros clientes o ser el contacto legal o apoderado de otros clientes.



Jerarquía de Clientes / Sedes

Los principales atributos que conforman al cliente son los siguientes:

- Datos personales: Nombre, Apellidos, Tipo y Número de Documento, Nacionalidad,…
- Tipología / clasificación: Clase, Tipo, Subtipo, Tratamiento, Mercado,…
- Perfil de valor: Segmento de valor, Fraude, Impago…
- Datos de contacto: Teléfono, Fax, Email, Idioma de contacto, Horario de contacto, Canal preferido,…
- Perfil sociodemográfico: Situación laboral, Profesión, Estado civil, Fecha de nacimiento,…
- Perfil comercial: Facturación, Beneficio, VIP…
- Marcas: No tratamiento de datos, No recepción publicidad,…

- Datos de Sistema: Identificador interno, Identificadores de otros sistemas, Fecha de alta en el sistema, Fecha de última modificación,…
- Estado del cliente

3.2.2 Sede

Es la entidad que muestra la estructura organizativa de un cliente empresa, identificando la ubicación de los distintos servicios o productos contratados. Son entidades informativas de la estructura organizativa de un cliente para criterios comerciales y asociación de servicios/productos. No se requiere la creación de estas estructuras si no se considera oportuno aunque suele ser habitual para ciertos tipos de clientes empresas.

La relación entre el punto de prestación de un determinado servicio (sede) y el punto de facturación (cuenta de facturación) se establece a través del servicio contratado.

El concepto de sedes no tiene vinculación directa con la estructura de facturación.

Los servicios contratados se pueden asociar a sedes de un cliente para poder conocer que sedes son las que contratan los servicios. Es una vinculación de carácter informativo sin repercusiones en facturación pero importante para la provisión o prestación de servicios para poder acometer actividades en una localización física.

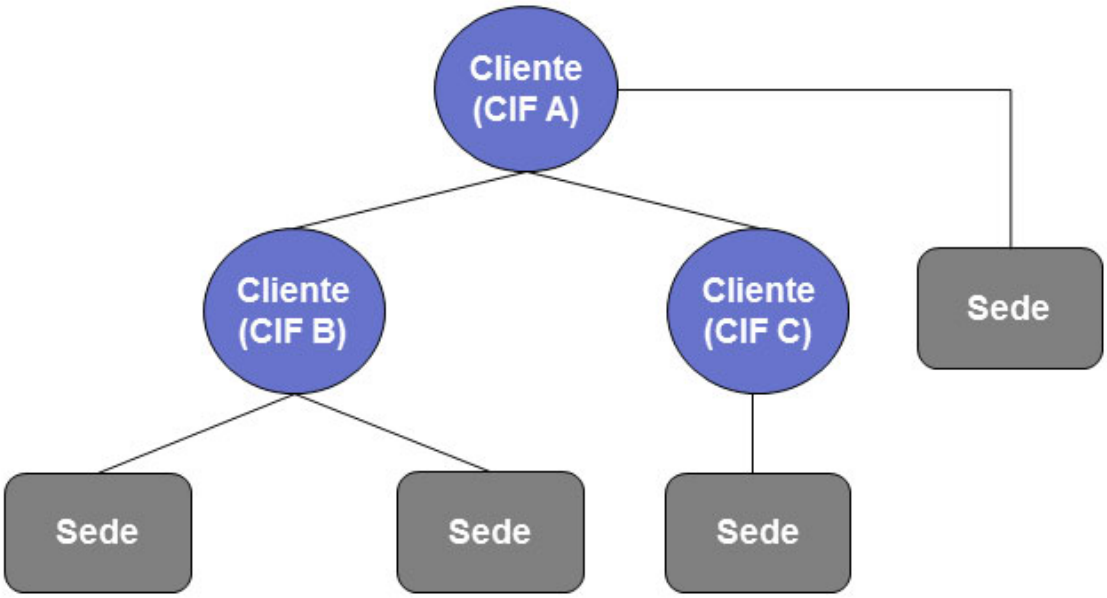
Cada Sede es única para un cliente. No puede ser reutilizada por otros clientes.

Un cliente podrá tener tantas Sedes como desee. Deberá ser coherente con la organización de la empresa y dar información válida para posteriores acciones comerciales o acciones de postventa.

Los atributos principales de las sedes son:

- Identificador: Nombre de la Sede o Departamento
- Tipo de Sede
- Direcciones asociadas
- Cliente y Personas de contacto a los que está asociada la sede
- Opciones de compra: Indica si la sede puede realizar compras, puede recibir facturas o puede recibir envíos.

- Datos del sistema: Identificador interno, Fecha de alta en el sistema, Fecha de última modificación,…



Estructura organizativa de un cliente empresa que identifica la ubicación de los distintos servicios contratados

3.2.3 Persona de contacto

Es la persona física asociada al cliente con la que la empresa que ofrece los productos/servicios interacciona.

Es posible establecer diversas jerarquías de personas de contacto dentro de un mismo cliente.

En el caso de empresas se pueden asociar personas de contactos a las sedes a la vez que asociarlos a los clientes.

Una persona de contacto puede tener varios perfiles o tipología.

Dentro de un mismo cliente únicamente puede existir una persona de contacto principal. Sin embargo en otras entidades asociadas con el cliente, como Sede y Cuenta Facturación, pueden tener sus personas de contacto principales.

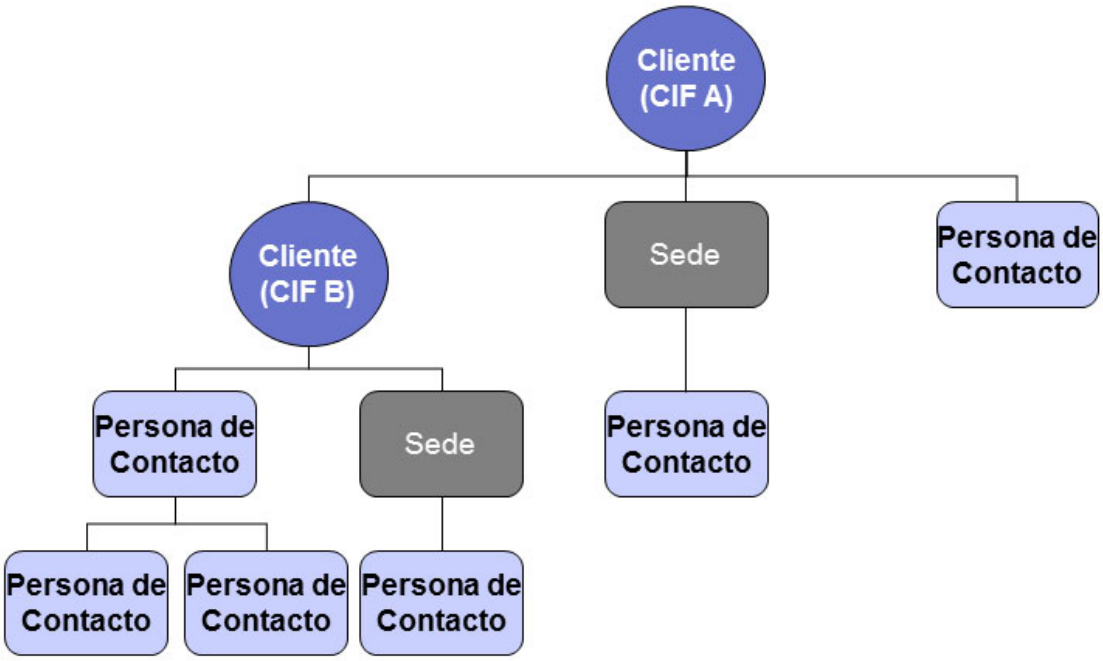
La tipificación de una persona de contacto se realiza a través de un conjunto de propiedades. De esta manera se permite que una misma persona pueda desempeñar varios roles al mismo tiempo, por ejemplo: contacto principal, apoderado, técnico, etc.

Cuando dos clientes comparten la misma persona de contacto, dentro del CRM se generan dos personas de contacto con perfil de contacto distinto. De este modo se posibilita la creación de perfiles de contacto diferentes para el mismo contacto de dos clientes distintos.

Una misma persona de contacto puede pertenecer a más de una entidad (sede, cuenta facturación) dentro de una relación de entidades encabezada por el cliente. Si se solicita la creación de una nueva persona de contacto en una entidad que no sea la del cliente, esta persona de contacto no queda relacionada directamente con el cliente, es decir, no hay una jerarquía directa, pero sí está relacionada a través de la entidad a la que se le ha creado.

Los atributos fundamentales de las personas de contacto son:

- Información personal: nombre, apellidos, tipo y número de documento, estado civil, sexo, fecha de nacimiento, nacionalidad, tratamiento, título, …
- Datos de contacto: teléfonos, email, fax, canal de contacto preferido, …
- Direcciones asociadas
- Tipología: Principal, autorizado, factura, técnico, titular,…
- Marcas: No email, No llamada, No SMS, No correo



Relaciones de Persona de Contacto

3.2.4 Dirección

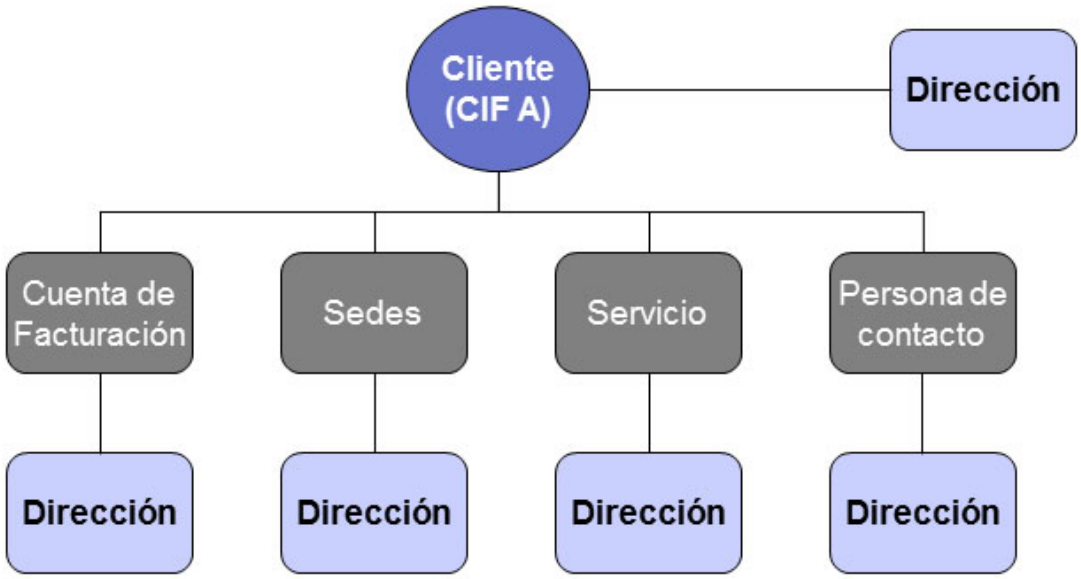
Es la ubicación física de la entidad con la que se relaciona: cliente, sede, cuenta, servicio, etc.

Dentro del CRM existe un repositorio de direcciones que se va alimentando a partir de las altas de nuevas direcciones que los clientes o los diferentes procesos de negocio van recogiendo.

Cada una de las entidades puede tener múltiples direcciones, pero siempre se asocia una de ellas como dirección principal. La propiedad de dirección principal es única para cada nivel de entidad de Cliente, Cuenta de Facturación y Sede. Las tres entidades no o tienen por qué tener la misma dirección principal.

Los atributos principales de las direcciones son:

- Información de la Dirección: tipo de vía, dirección, número, escalera, piso, puerta, localidad, provincia, código postal, país
- Tipología: principal, envío, contrato,…
- Estado: Activa / Inactiva



Relación de Direcciones

3.2.5 Interacción

Se entiende por interacción como cualquier comunicación multicanal entrante o saliente mantenida entre el cliente, la empresa proveedora de productos/servicios y los canales de venta/contratación.

Las interacciones se registran asociadas al cliente o a la persona de contacto.

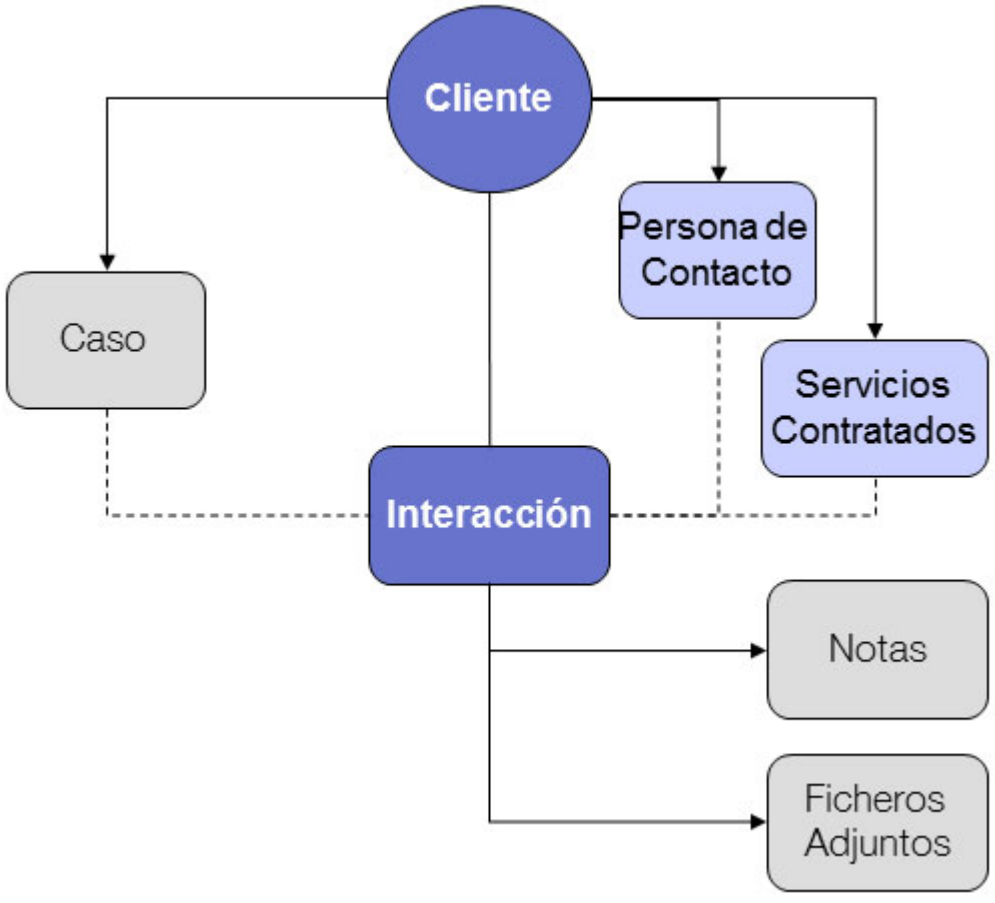
Cuando una interacción surge como una petición de cliente entonces dicha interacción lleva asociada una solicitud (caso) para hacer gestión y seguimiento de la petición.

Se suelen tratar como interacciones entrantes aquellos contactos con el cliente a través de los canales que la empresa considere oportuno habilitar, por ejemplo: teléfono, email, SMS, IVR (Interactive Voice Response).

Los atributos fundamentales de las interacciones son:

- Tipo: Interacción Saliente, Interacción Entrante, Encuesta, Incidencia...
- Cliente asociado
- Servicio contratado / Producto adquirido
- Método de Contacto: Teléfono, IVR, Fax, Email, SMS…
- Usuario y Grupo Funcional

- Tipificación: Tipo/Familia/Categoría/Subcategoría
- Estado, Fecha de Estado
- Entidades relacionadas: Caso, Notas y Anexos.
- Información de sistema: Identificador interno, fecha de creación, usuario creador, fecha de modificación, fecha de cierre,…
- Información adicional propia de cada tipo: Número llamante, Mensaje SMS, Teléfono SMS
- Tipo de solución: on-line, off-line



Relación de Interacción con resto de entidades

3.2.6 Caso

Es la entidad utilizada para registrar un problema asociado a la prestación y uso de los productos o servicios contratados por el cliente y a la atención al mismo. Así mismo un caso también puede ser utilizado para la gestión de peticiones, devolución de material, solicitud de información, etc.

El caso tiene un determinado ciclo de vida y pasa por diversos estados y grupos de proveedores (agrupación de operadores del CRM) que realizarán las acciones oportunas para su tratamiento y resolución.

El ciclo de vida del caso empieza por la apertura del mismo por parte de un agente. Dicho agente tipificará el caso en base a cuatro valores fundamentales: Tipo de Caso, Familia del producto objeto del caso, Categoría y Subcategoría. Estos valores son configurables y administrables dentro del CRM para adecuarse a las necesidades de la empresa. Además se pueden establecer filtros en base a los valores que van a elegir en los desplegables de tipología de jerarquía superior (por ejemplo según la categoría que se seleccione, se pueden mostrar unas subcategorías u otras).

Una vez tipificado el caso, el agente también podrá registrar otra información necesaria para su resolución en forma de notas y anexos y podrá asignar el caso a un agente concreto o a un grupo de proveedores.

Un grupo de proveedores lo componen un conjunto de agentes con unas características comunes y con los permisos necesarios para gestionar unos determinados casos. Cuando un agente del grupo de proveedores coge el caso, hará las acciones pertinentes para su resolución y registrará en el caso aquello que ha realizado. Este agente podrá modificar el estado del caso y consultar los estados por los que ha pasado así como relacionar el caso con otros objetos como por ejemplo otro caso, crear notas y anexos o asignar el caso a otro agente o grupo de proveedores.

Es habitual que el CRM permita configurar un proceso de envío de notificaciones para informar a la lista de trabajo del grupo de proveedores la asignación de un nuevo caso. Este es el caso de PeopleSoft CRM.

Una vez tratado el caso el agente podrá proceder a cerrar el caso. Para ello seleccionará el estado Cerrado en el desplegable de Estado del Caso. Los estados del caso son valores configurables pero siempre se agrupan en las categorías: Abierto, Cerrado o Cancelado. Para cerrar el caso es obligatorio asignar una solución de cierre.

Las soluciones son maneras en las que se puede resolver el caso. Si la solución se puede aplicar de forma satisfactoria entonces se podrá poner como solución correcta. El estado del caso es independiente de la solución por lo que un caso podría estar solucionado y permanecer en estado abierto.

Dentro del objeto caso se pueden ver las soluciones intentadas.

Relación del caso con otros objetos

Los atributos fundamentales de los casos son:

- Tipificación: Tipo/Familia/Categoría/Subcategoría.
- Cliente asociado
- Servicio contratado
- Cuenta asociada.
- Persona de contacto
- Usuario y/o Grupo Funcional
- Soluciones
- Estado, Fecha de Estado
- Entidades relacionadas: Otros casos, Notas y Anexos.
- Información de sistema: Identificador interno, fecha de creación, usuario creador, fecha de modificación, fecha de cierre,…

3.2.7 Cuestionario

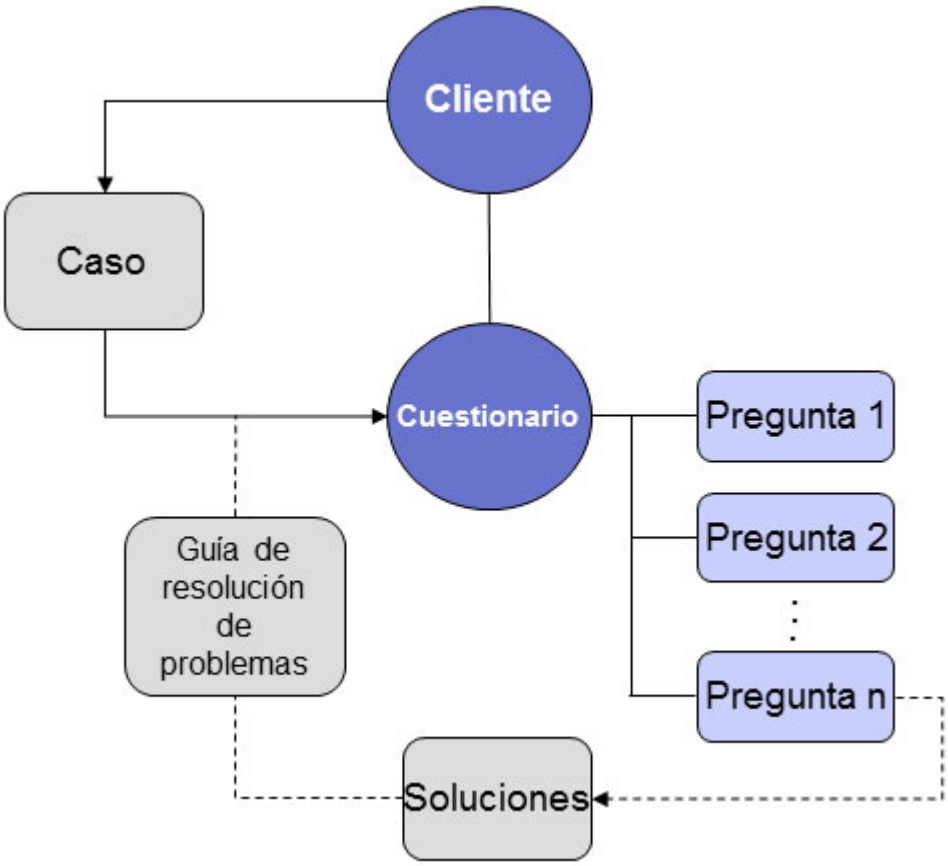
Los cuestionarios son un conjunto de preguntas que se efectúan al cliente.

La principal utilidad de los cuestionares es la recopilación de información sobre el problema del cliente para solucionar el caso, aunque también es posible parametrizar cuestionarios para realizar encuestas de satisfacción o cualquier otra actividad que conlleve recabar información del cliente.

Los cuestionarios que se para recabar información y resolver casos se denominan en PeopleSoft CRM con el nombre de “Guía de solución de problemas” y están ligados directamente con el caso. En base a la recopilación de información se puede configurar el cuestionario para ofrecer soluciones al problema o transferir al agente a páginas de soluciones.

Habitualmente se predetermina las respuestas del cliente al cuestionario mediante respuestas acotadas en las que se elija un posible valor pero también existe la posibilidad de establecer respuestas en modo texto libre. La elección de respuestas en modo texto libre es una opción poco utilizada ya que la potencia real del cuestionario radica en la parametrización

de acciones en base a la respuesta del cliente o la explotación de la información en sistemas de Datawarehouse.



Relación del cuestionario con otros objetos

Los atributos fundamentales de los cuestionarios son:

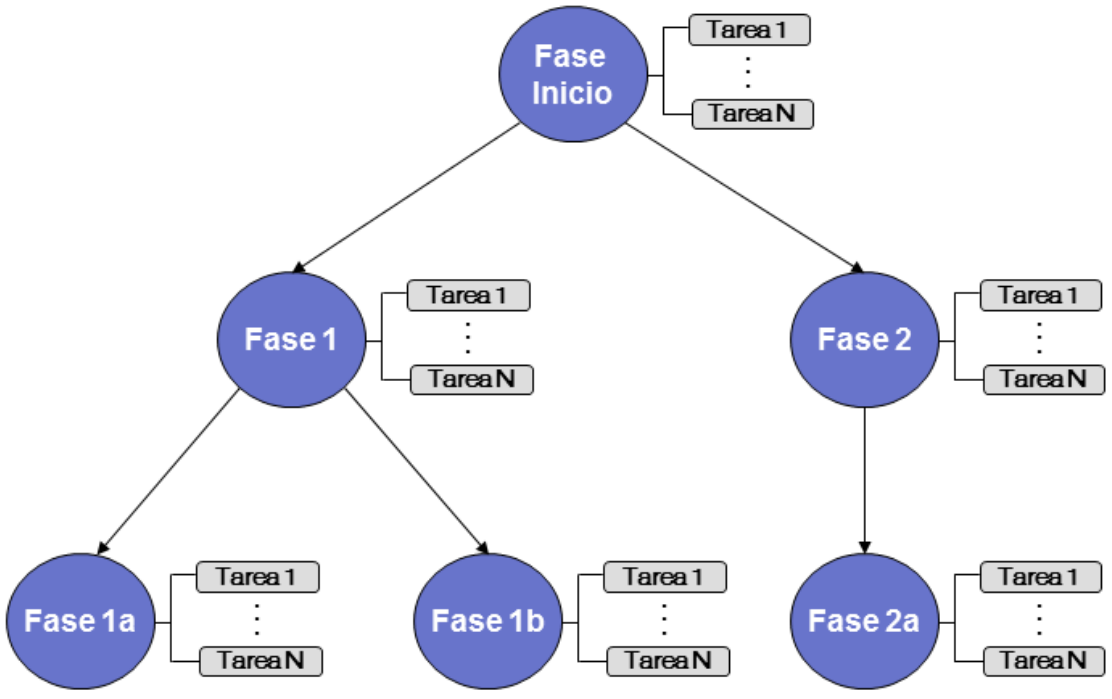
- Tipo: encuesta, resolución problema, etc
- Preguntas (1..N)
- Respuestas a preguntas: valores acotados o texto libre
- Cliente asociado
- Caso asociado
- Soluciones posibles
- Estado, Fecha de Estado

- Información de sistema: Identificador interno, fecha de creación, usuario creador, fecha de modificación, fecha de cierre,…

3.2.8 Proyecto de Gestión

El proyecto de gestión es una entidad propia de PeopleSoft CRM que por su importancia a la hora de la gestión y resolución de casos merece una mención específica.

Un proyecto de gestión es un workflow o flujo de trabajo estructurado con la finalidad de coordinar tareas. Permiten establecer árboles de decisiones complejos con reglas de transición. Los proyectos de gestión se componen de fases y cada una de estas fase de 1 o n tareas.



Proyecto de Gestión

La utilización de proyectos de gestión conlleva las siguientes ventajas:

- Permite poder trabajar de forma ordenada en modo plantilla con las soluciones más efectivas
- Permite secuenciar las acciones y por tanto garantizar la realización de las tareas en el orden correcto.

- Permite automatizar notificaciones y procesos en base al estado de las fases y tareas.

Cuando se instancia un caso, se puede crear un proyecto de gestión de forma manual asociado a dicho o caso o bien se puede parametrizar que dicho proyecto de gestión se instancie de forma automática con la creación de caso.

La transición entre las fases del proyecto de gestión la puede hacer el operario que esté gestionando el caso de forma manual o también se puede realizar de forma automática en base al estado de la finalización de la tareas, pudiendo administrarse reglas de asignación para el tránsito a una fase u otra.

Cada una de las fases del proyecto de gestión podrá estar paremetrizada para que le llegue la tarea a un determinado grupo de proveedores y que así puedan actuar y resolver las mismas.

El proyecto de gestión es un objeto muy potente de PeopleSoft CRM que permite una gran ventaja respecto a otros CRM comerciales para la simplificación y correcta gestión de la atención al cliente.

3.3 Caso de uso: atención de un caso tipo de cliente

Para explicar los conceptos que hemos visto vamos a proponer un caso de uso tipo en el que se explique cómo sería la gestión en el CRM de un cliente que tiene una incidencia con un hipotético servicio contratado a una empresa.

Supongamos que el ***cliente*** llama al número de atención al cliente por una incidencia que afecta a su servicio contratado. Lo habitual es que el cliente sea atendido por un sistema ICM (Intelligent Contact Management) e IVR (Interactive Voice Response). El sistema IVR será el encargado de ofrecer locuciones telefónicas al cliente para guiarle en un árbol de decisión y el sistema ICM será el encargado de establecer la inteligencia en los procesos para interactuar con otros sistemas y hacer el enrutamiento del mismo. Habitualmente el sistema ICM estará integrado con el CRM, en primera instancia para saber si el usuario que llama es cliente o no y para dejar reflejo en el CRM de la gestión realizada.

Si tras navegar por el árbol de decisiones el cliente ha encontrado ha encontrado una ***solución*** al problema, entonces se creará una ***interacción*** en el CRM asociada al cliente para dejar reflejo de que se ha establecido un contacto con el cliente y la solución aplicada.

En el supuesto de no encontrar solución el sistema ICM enrutará la llamada a un agente que tenga el aplicativo CRM para realizar las gestiones oportunas. Es habitual que las grandes compañías tengan distintos Call Center (centros de atención al cliente) distribuidos incluso geográficamente y especializados en distintas gestiones; por ejemplo, se puede tener un Call Center especializado en problemas relacionados con la facturación, otro que atienda los problemas de envíos logísticos y otro que atienda las incidencias técnicas de los clientes. En cada uno de estos Call Center los agentes accederán al sistema CRM para realizar las gestiones.

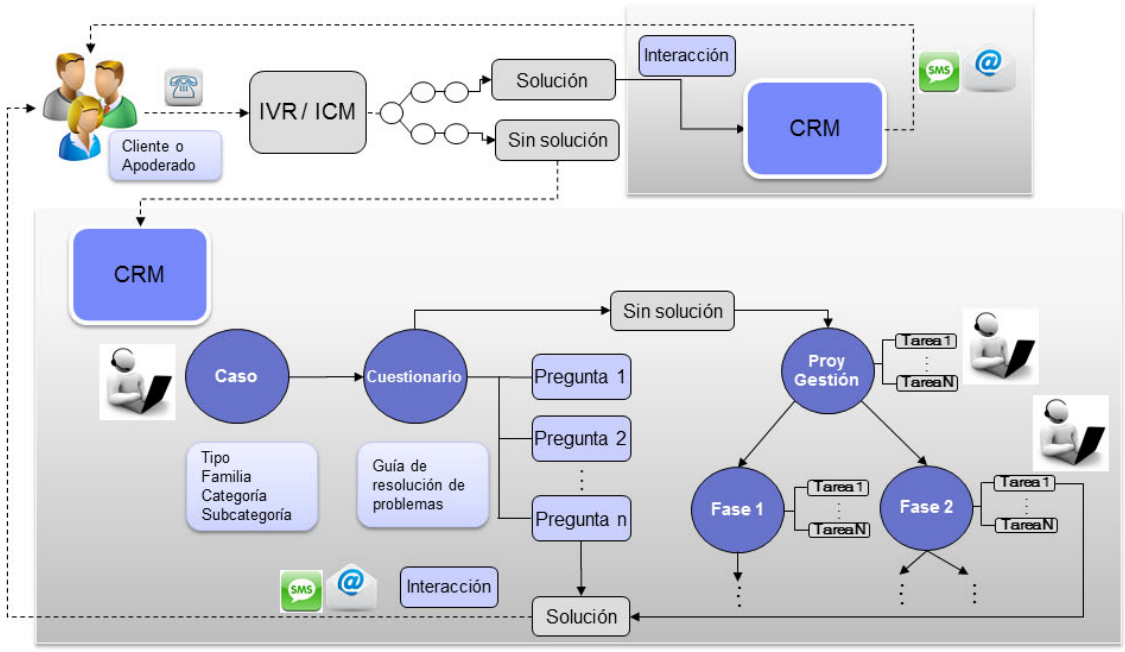
Cuando el agente recibe la llamada transferida por ICM, lo primero que hará será hacer comprobaciones sobre el ***cliente***. Por ejemplo, se puede buscar al ***cliente*** por documento de identidad y realizar alguna pregunta sobre su nombre y apellidos así como fecha de nacimiento para validar que estamos hablando realmente con el cliente. También existe la posibilidad de que el llamante no sea el propio cliente sino una ***persona de contacto*** autorizada. En este caso, puesto que el CRM tiene la entidad ***persona de contacto***, se podrá realizar una gestión adecuada del problema.

Tras identificar al ***cliente*** y validar su identidad el agente del Call Center creará un ***caso*** para categorizar el problema. Para la creación del ***caso*** seleccionará de los desplegables habilitados el ***tipo de caso*** (en este supuesto sería de tipo incidencia), la ***familia*** del servicio contratado, la ***categoría*** y la ***subcategoría***. La creación del ***caso*** instanciará automáticamente un ***cuestionario*** y un ***proyecto de gestión*** asociados al mismo.

El agente realizará las preguntas del cuestionario al ***cliente*** para encontrar una solución. Es lo que se conoce como una ***guía de resolución de problemas***. Si encuentra una solución válida procederá a aplicar la misma y cerrar el ***caso***. Además quedará reflejada una ***interacción*** con el ***cliente*** y se podrá enviar un email o sms para informar al ***cliente*** de la resolución de la incidencia.

En el supuesto de no encontrar la solución, el agente despedirá al ***cliente*** y le indicará que su incidencia se tratará en backoffice y que se le informará de la resolución de la misma. Tras despedir al ***cliente***, el agente accederá en el CRM al ***proyecto de gestión*** asociado al ***caso*** y lo transitará a una ***fase*** determinada para que lo atienda un equipo de backoffice (***grupo de proveedores*** de backoffice). Dicho caso llegará a la cola de gestión de un agente del grupo de backoffice y tratará de buscar ***solución*** al mismo realizando las ***tareas*** propias de la ***fase*** del ***proyecto de gestión***. Si no la encuentra podrá transitar el ***proyecto de gestión*** a otra ***fase*** para sea atendido por otro grupo según la necesidad añadiendo las ***notas*** y ***anexos*** que sean necesarios para facilitar la labor del grupo siguiente.

El agente del grupo que resuelva el problema procederá a cerrar el ***caso*** con la ***solución*** aplicada y se disparará automáticamente una ***interacción*** saliente al ***cliente*** vía SMS o email para indicar la resolución de la incidencia.



Caso de uso: atención a cliente con incidencia

4 ARQUITECTURA DE PEOPLESOFT

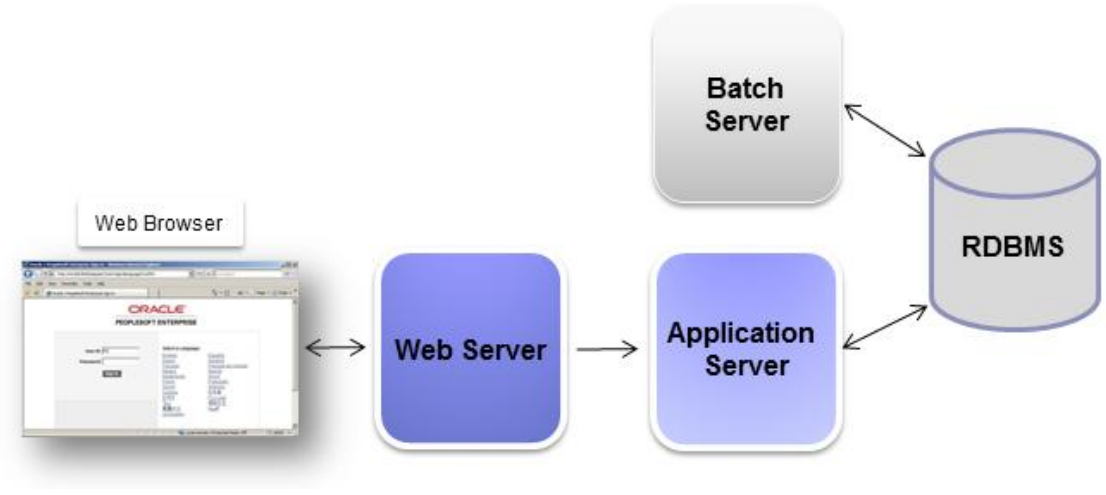
4.1 Conceptos básicos de PeopleSoft Internet Architecture (PIA)

La arquitectura de PeopleSoft es conocida por el nombre *"PeopleSoft Internet Architecture"* (PIA). En sus orígenes PeopleSoft tenía una arquitectura pura cliente-servidor, sin embargo, con el paso del tiempo esta arquitectura evolucionó a una arquitectura orientada a la Web apareciendo el concepto de PIA dentro del mundo PeopleSoft. La arquitectura está formada por los siguientes componentes:

- Web Browsers: navegadores web
- Web Server(s): servidor web
- Application Server(s): servidor de aplicaciones.
- Process Scheduler Server(s): servidor de procesos batch.
- Sistema gestor de base de datos relaciones (RDBMS)

En la siguiente imagen se muestran los componentes de la arquitectura de PeopleSoft a alto nivel. Un navegador se conecta con un servidor web (Web Server) que pueda ejecutar servlets. Para ello se instala una colección de servlets de PeopleSoft diseñados para manejar las transacciones. El servidor web se conecta mediante JOLT (un protocolo de Tuxedo) con el Application Server, que es el centro de la arquitectura: ejecuta la lógica de negocio y realiza transacciones SQL con el servidor RDBMS. Este Application Server consiste en un numeroso conjunto de servicios PeopleSoft y procesos del servidor que manejan peticiones de transacción del navegador. La comunicación que realiza el Application Server con el navegador es con el envío de HTML, JavaScript y Cookies.

PeopleSoft usa TUXEDO (Transactions for Unix, Extended for Distributed Operations), un framework middleware y monitor de transacciones, para gestionar las transacciones de la base de datos, y JOLT, que es un API Java, para facilitar las peticiones de transacciones desde Internet. Ambos productos pertenecían a BEA Systems que también fue adquirida por Oracle.



PeopleSoft Internet Architecture (PIA)

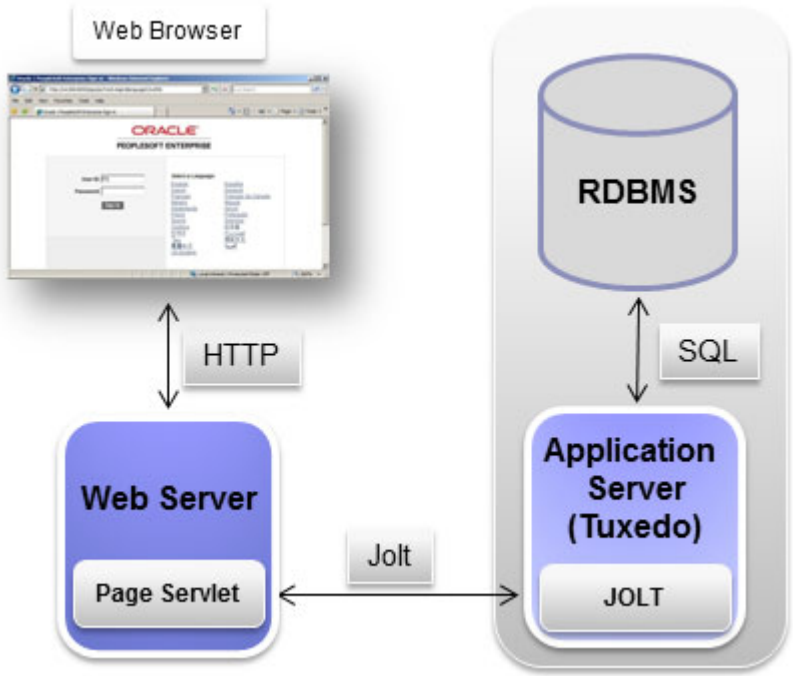
El servidor BD contiene toda la base de datos PeopleSoft, incluyendo todas las definiciones de objetos, las tablas de sistema, las tablas de aplicación y los datos. El servidor de bases de datos se ejecuta en una combinación RDBMS/sistema operativo.

La relación entre el servidor de BD y el Application Server es 1 a N. Un único servidor de BD puede tener varios Application Servers conectados a él. El servidor BD maneja simultáneamente las conexiones del Application Server, conexiones del entorno de desarrollo (Application Designer) y los programas batch corriendo contra él.

El servidor batch es el servidor en el que se tiene instalado y configurado el Process Scheduler, y es el encargado de ejecutar los procesos batch tales como los programas Application Engine, o SQR (un tipo específico de programas de PeopleSoft).

En la instalación PeopleSoft, se instalan varios servlets en el Web Server. Además del software servidor de web, por tanto, se necesita un motor de servlets instalado, como Jserv.

En la siguiente imagen vemos como se comunica el servlet que ejecuta el servidor web con el servidor de aplicación mediante la capa JOLT. JOLT es la capa de comunicación entre Java y el entorno en el que se ejecuta Tuxedo. Como se ve, Jolt es inherente a Tuxedo: debe coexistir con él, no puede funcionar solo.



PeopleSoft Internet Architecture (PIA)

4.2 Application Server Domains

Se denomina así a la colección de Procesos de Servidor, procesos de soporte y gestores de recursos que habilitan conexiones a la base de datos. Cada Application Server Domain tiene un fichero de configuración, y se configura cada servidor de aplicación para conectarse a una única BD. Una máquina que actúe de Application Server puede soportar varios Application Server Domains corriendo. Para configurar un Application Server Domain se usa la herramienta PSADMIN, localizada en el directorio $PS_HOME/appserv de la instalación de PeopleSoft.

Podemos conectar un único Application Server Domain a una base de datos, o bien, varios Application Server Domains a una sola base de datos.

Por ejemplo, supongamos que tenemos 3 bases de datos: BBDD01, BBDD02 y BBDD03. En ese caso necesitaríamos 3 Application Server Domains, uno por cada BD. Según el aumento de la demanda, quizá se necesitaría configurar varios Application Server Domains

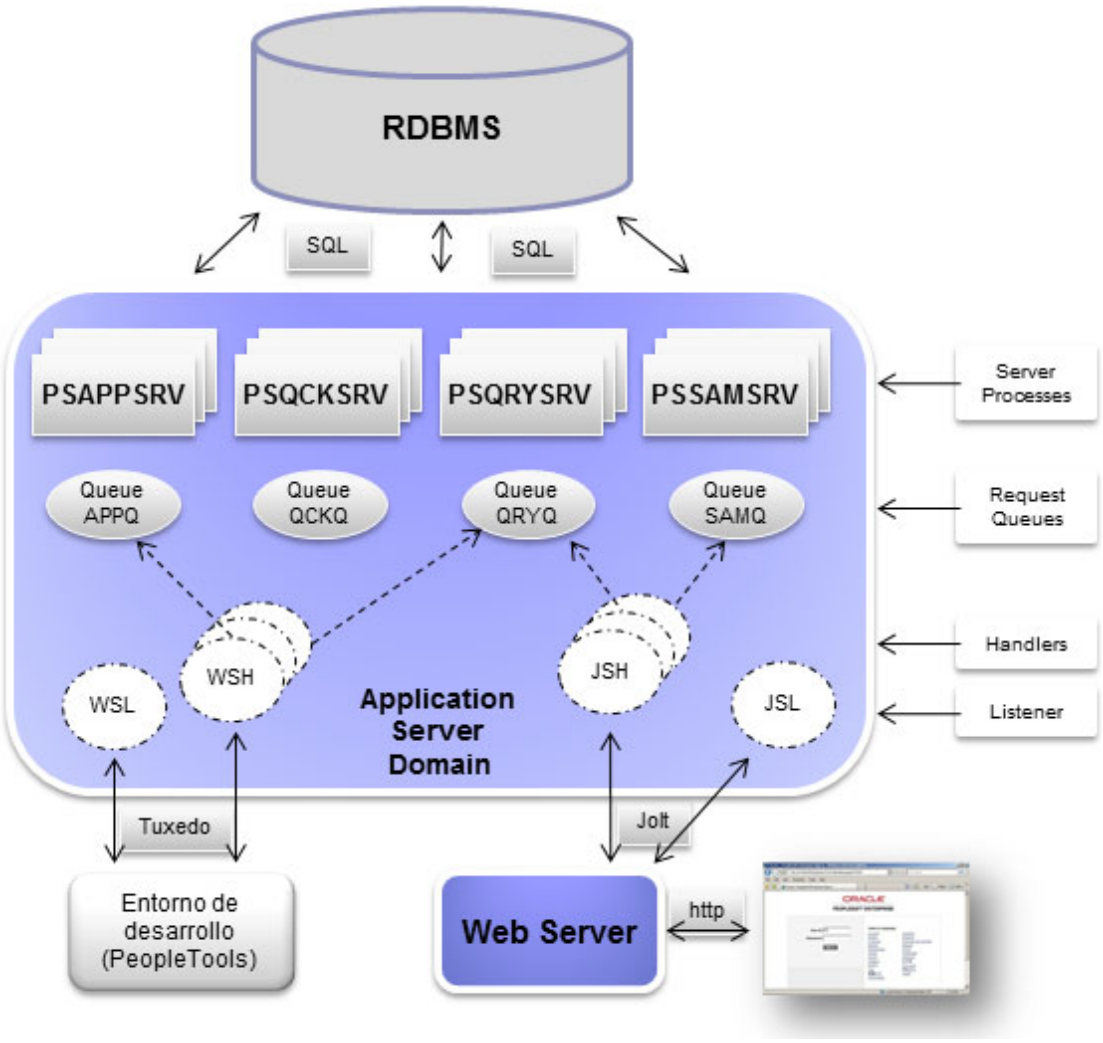
por cada base de datos, para obtener un grado de redundancia y tolerancia a fallos, así como un cierto nivel de rendimiento.

Los Application Server Domains se configuran bajo un único directorio HOME de PeopleSoft (PS_HOME, que es el directorio destino de la instalación del Application Server). PSADMIN crea un directorio bajo PS_HOME por cada Application Server Domain. Teniendo en cuenta el ejemplo anterior:

```
\PS_HOME
   \APPSERV
      ...
      \BBDD01
      \BBDD02
      \BBDD03
      ...
```

Cuando se arranca el Application Server Domain, se arrancan procesos de servidor asociados al dominio como PSAPPSRV, PSQCKSRV, PSSAMSRV, etc. Cada proceso de servidor establece una conexión persistente con la BD PeopleSoft, que usará como una tubería (pipe) SQL por la que el proceso servidor enviará y recibirá SQL. Cada proceso del servidor usa una única conexión SQL para servir peticiones de diferentes fuentes. Desde el punto de vista de la BD, cada proceso del servidor es un solo usuario.

Para explicar los procesos del servidor en detalle utilizaremos la siguiente figura:



Procesos de servidor

4.2.1 Workstation Listeners (WSL) y Workstation Handlers (WSH)

WSL es el punto inicial de contacto de las equipos/entornos de desarrollo con el servidor. Después de recibir la petición de transacción, la pasa al proceso WSH. En este momento, el entorno de desarrollo solo se comunica con su Workstation Handler asignado.

WSL también monitoriza la distribución de conexiones con las WSH, de manera que cuando el número de conexiones aumenta por encima de un umbral, WSL arranca nuevos WSH para satisfacer la demanda.

WSH recibe la petición de transacción de WSL y maneja la transacción de la Workstation. Es posible aumentar el pool de procesos WSH para gestionar mayor carga según aumente la demanda.

4.2.2 Jolt Server Listener (JSL) y Jolt Server Handler (JSH)

El JSL maneja las peticiones del browser (a través del Servlet que se ejecuta en el Web Server). Escucha las peticiones del browser y entrega la conexión a un proceso JSH que esté disponible.

El proceso JSH gestiona las transacciones que han sido pedidas por el usuario desde el browser. Gestiona la conectividad de red, y se encarga del diálogo con el servidor de aplicaciones (Tuxedo) mediante JOLT.

4.2.3 Request Queues (APPQ, QCKQ, QRYQ, SAMQ)

Cada proceso del servidor tiene una cola de peticiones de servicio que comparte con otros servidores del mismo tipo (por ejemplo, la cola APPQ para los procesos PSAPPSRV). Los procesos WSH y JSH insertan peticiones en la cola apropiada, de forma que los procesos que las atienden lo hagan en orden de llegada. La imagen anterior indica un solo WSL y JSL, y varios WSH, JSH y procesos de servidor. Hay una sola cola por cada tipo de proceso de servidor.

4.2.4 PSAPPSRV

PSAPPSRV es el proceso principal de servidor que corre dentro del Application Server Domain. Su misión es ejecutar las peticiones funcionales como la construcción y carga de componentes. Gestiona también la memoria y la cache de los objetos PeopleTools en el servidor de aplicación. Cada uno de estos procesos tiene su propia memoria y cache de disco. También proporciona los servicios de autenticación para los usuarios que quieren entrar en el sistema. Es un proceso obligatorio.

4.2.5 PSQCKSRV

Es en esencia una copia del proceso PSAPPSRV. Se encarga de realizar peticiones SQL rápidas y de sólo lectura. Es un proceso opcional, diseñado para mejorar el rendimiento del sistema, ya que se encarga de manejar algunos elementos de la cola de peticiones al proceso PSAPPSRV un proceso opcional.

4.2.6 PSQRYSRV

De la misma manera que el proceso PSQCKSRV, PSQRYSRV se ha diseñado para aliviar la carga de PSAPPSRV. Se utiliza para manejar todas las peticiones generadas por el usuario mediante el programa PeopleSoft Query (PSQED.EXE). Este proceso de servidor mejora el rendimiento medio del servidor de aplicaciones, este o no configurado PSQCKSRV. Está específicamente diseñado para procesar transacciones SQL de PeopleSoft Query, que pueden ser muy intensivas. Es un proceso opcional.

4.2.7 PSSAMSRV

Este proceso está diseñado para procesar todas las transacciones SQL que provienen del programa Application Designer. Es un proceso obligatorio para cualquier dominio.

4.3 Servidor batch: Process Scheduler

En el entorno PeopleSoft, el servidor batch se denomina Process Scheduler. Típicamente es el servidor en el que corren los programas Application Engine y otros programas batch. Se puede instalar el Process Scheduler en un servidor dedicado, o bien correr en el Application Server o en el Database Server. Process Scheduler monitoriza la tabla estándar de Bases de Datos de procesos PeopleSoft (PSPRCSRQST) y arranca Application Engines, COBOL, SQR y otros procesos batch.

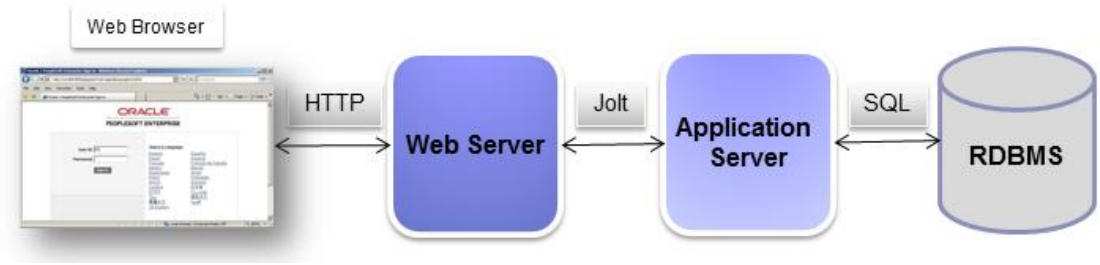
PeopleSoft usa PSADMIN para configurar tanto el Process Scheduler como el Application Server . Se puede instalar Process Scheduler en una máquina que soporte una base de datos compatible con PeopleSoft, pero no compatible con un servidor de aplicación. En tal caso, se puede usar PSADMIN para configurar el Process Scheduler, pero las opciones de

menú que hagan referencia al Application Server no funcionarían al no tener compatibilidad con el servidor de aplicación. Esto sirve para ilustrar, cómo PeopleSoft usa PSADMIN para la configuración tanto de Process Scheduler como del Application Server. Ambos comparten directorios bajo $PS_HOME. Aun así son entidades separadas que se arrancan, configuran y paran de manera separada.

4.4 Configuraciones físicas típicas

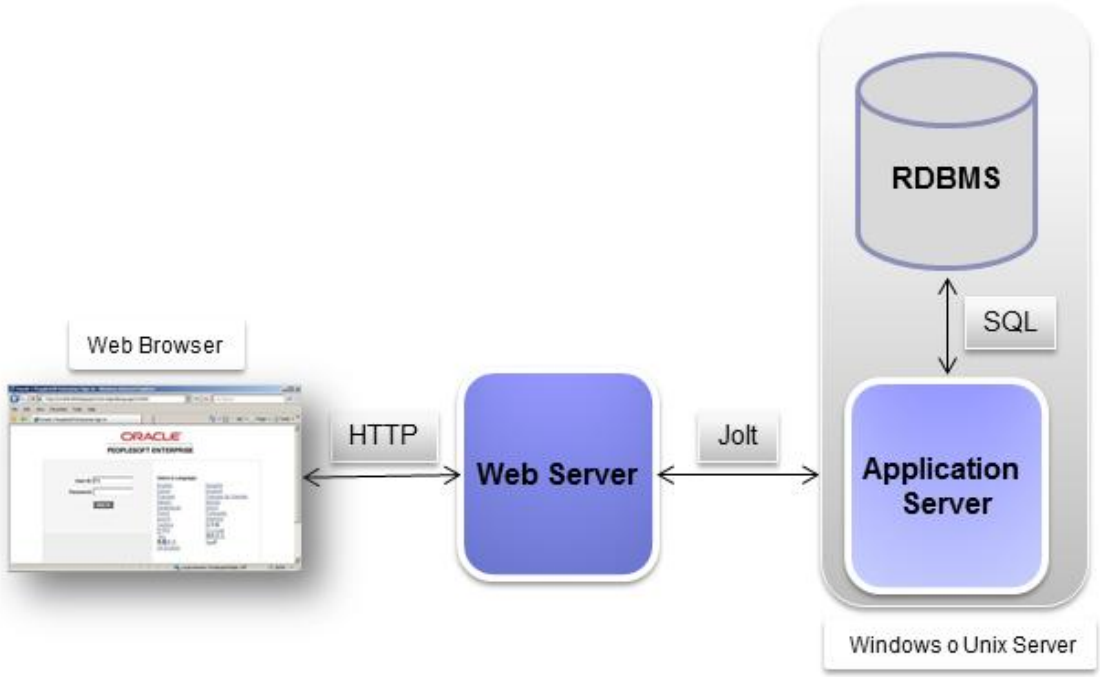
Hay una serie de configuraciones típicas que se pueden utilizar para distribuir los componentes de la arquitectura PeopleSoft en diferentes máquinas. Las configuraciones típicas son:

1) Cada componente en una máquina: una para el servidor web, una para el Application Server y otra para el servidor de base de datos.



Configuración de componentes separada

2) O bien, el servidor web en una máquina y el servidor de aplicaciones y el servidor de BD juntos en otra máquina:



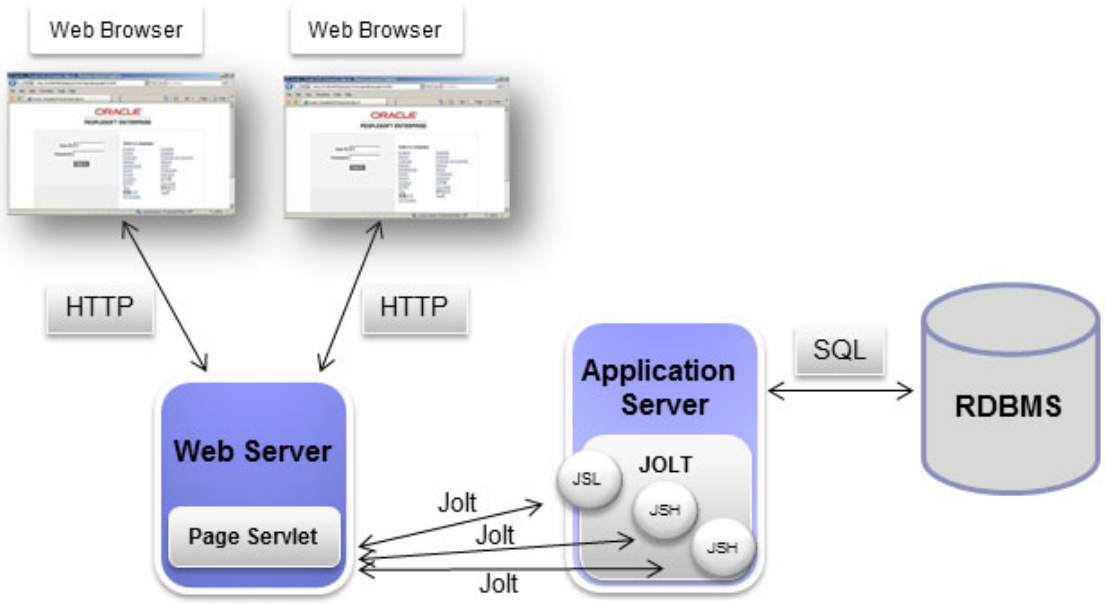
Configuración de componentes agrupada

3) O bien podría estar el Web Server en la misma máquina con el servidor de aplicaciones y servidor de base de datos. Aunque esta arquitectura es factible, se desaconseja por motivos de seguridad ya que el Web Server debería estar solo, tras un firewall, si se va a abrir la aplicación a Internet de forma pública.

4.5 Opciones para el despliegue

4.5.1 Conexión a través de PIA

Mediante la conectiva PIA un browser se conecta con el servidor web a través del servlet. Las interacciones browser – servidor web son siempre con HTML.



Conexiones PIA

Todo el procesamiento ocurre en el Web Server y en el Application Server. En el servidor web, el Page Servlet formatea el HTML y gestiona la conexión con el proceso JSL vía JOLT. Posteriormente, el Page Servlet enruta la petición del browser a un JSH disponible. Este pasa la petición al Application Server, que a su vez transmite la petición de SQL a la base de datos.

Hay tantos JSHs como browsers emitiendo peticiones, por tanto habrá un número igual de conexiones a las JSHs. Es posible realizarlo mediante una sola conexión pero en este caso podríamos experimentar problemas de rendimiento.

El proceso PSAPPSRV genera el HTML requerido para el browser y lo devuelve al JSH del que vino la petición. Para la base de datos, estas transacciones son simples transacciones del tipo Cliente – Servidor.

4.5.2 PeopleSoft Portal

Permite integrar contenidos PeopleSoft con otros contenidos provenientes de otras fuentes.

PeopleSoft Portal consiste en un Portal Servlet y un Application Server. Estos dos componentes trabajan juntos para proveer procesamiento común del portal, como uso de páginas, búsquedas, gestión de contenidos, etc.

4.5.3 Conexión desde el entorno de desarrollo

La conexión a PeopleTools no se realiza directamente contra el browser sino a través del aplicativo Application Designer. Dicha conectividad se puede establecer bien en dos capas, directamente contra la base de datos, o bien a través del servidor de aplicaciones (tres capas).

4.5.4 Tecnologías para la integración

Las tecnologías PeopleSoft disponibles para la integración con sistemas de terceros u otras bases de datos son las siguientes:

- Application Messaging: Arquitectura de mensajería Publish/Subscribe para integración asíncrona y sincronización de datos. Maneja mensajes entrantes y salientes. Esta arquitectura ha sido substituida por Integration Broker a partir de la versión 8.1x de PeopleTools aunque se mantiene por compatibilidad.

- PeopleSoft Integration Broker: Es una tecnología middleware SOA (Service Oriented Architecture) que facilita el intercambio de mensajes, tanto síncronos como asíncronos, entre sistemas internos o externos. En este intercambio es posible gestionar la estructura y el formato del mensaje, de forma que se produzcan las transformaciones en los mensajes necesarias entre sistemas durante el envío de los mismos.

  Este producto, como hemos comentado, sustituye a la plataforma Application Messaging de la versión de PeopleTools 8.1x.

  El Integration Broker se compone de dos subsistemas:

  - Integration Engine, que corre en el Application Server. Está acoplado a las aplicaciones PeopleSoft y produce y consume mensajes para estas aplicaciones.

  - Integration Gateway es una plataforma que funciona bajo un servidor web, que gestiona la recepción y entrega de los mensajes que se pasan entre los sistemas a través del Integration Broker. Proporciona soporte para los principales protocolos TCP/IP (HTTP, SMTP, POP3 y FTP), y más importante, proporciona un SDK para el desarrollo de nuevos conectores con aplicaciones heredadas, ERPs y sistemas o aplicativos basados en una arquitectura web.

- Component Interfaces: Arquitectura de componentes orientados a objetos, request/reply, que permite a aplicaciones de terceros invocar síncronamente la lógica de negocio PeopleSoft.

- Business Interlinks: Es un plug-in framework que permite a las aplicaciones PeopleSoft invocar APIs de terceros a través de Internet.

- File Layout: Es una clase de PeopleCode que proporciona métodos y propiedades para leer y escribir en ficheros. El File Layout como objeto es una definición (o mapeo) de un fichero a procesar, que elimina la necesidad de analizar sintácticamente el fichero, dividiéndolo en registros y campos y encapsulando toda la lógica de formatos, separadores de campos, tipos de datos, etc.

- Invocación de servicios web: PeopleCode permite el uso de *iScripts* que son funciones especializadas PeopleCode que generan contenido web de forma dinámica. Los iScripts interactúan con clientes web (browsers) usando un paradigma *request - response* basado en el comportamiento del protocolo HTTP. Por tanto, el uso de iScripts es similar al de los *servlets Java* y puede ser útil para integrar sistemas externos.

Mediante PeopleCode, a partir de PeopleTools 8.4, es posible invocar servicios externos SOAP. Por tanto, la integración con otros sistemas vía *web services* es posible en ambos sentidos (exponiendo funcionalidad de componentes vía SOAP mediante *component interfaces*, e invocando servicios web externos, también vía SOAP, desde PeopleCode).

5 MANUAL DE USUARIO: INTORDUCCIÓN A PEOPLESOFT

5.1 Visión General

La interfaz de usuario de PeopleSoft es una interfaz de tipo web. El usuario accede a la URL de conexión mediante un navegador (Firefox, Chrome, IE Explorer, etc) y se conecta con su usuario/contraseña según la política de accesos definida. También es posible logarse en el sistema mediante un dispositivo portable como una Tablet o SmartPhone, pero siempre será mediante un navegador web estándar.

En este capítulo se detallan los conceptos básicos para facilitar la comprensión del sistema y se explican las partes más importantes de la interfaz de cara a que un nuevo usuario pueda entender los conceptos claves sobre el uso y navegación en PeopleSoft.

5.2 Conceptos básicos de navegación

5.2.1 Menú lateral

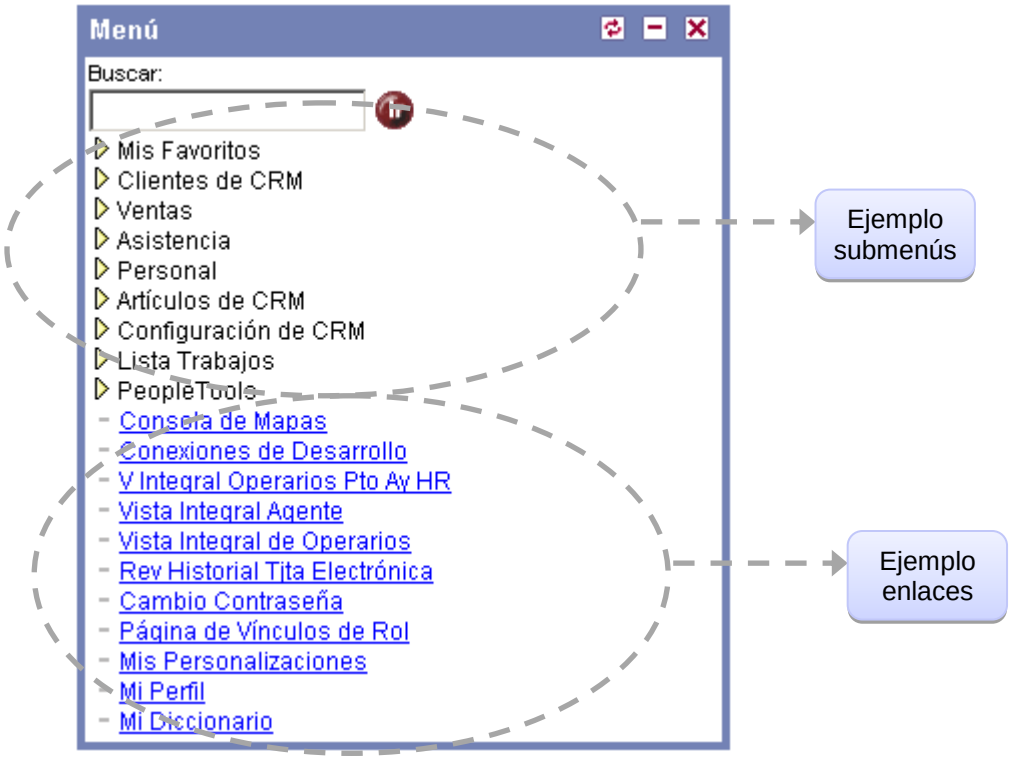
El acceso a las diferentes secciones del sistema se realiza mediante un menú situado a la izquierda de la pantalla.

Dentro del menú debemos distinguir entre menús desplegables y enlaces.

- Menús desplegables: Permiten acceder a otras opciones de menú, y tienen estructura jerárquica.
- Enlaces: Permiten navegar a pantallas del sistema.

Si pinchamos sobre un menú desplegable, aparecerán nuevas opciones de menú o enlaces.

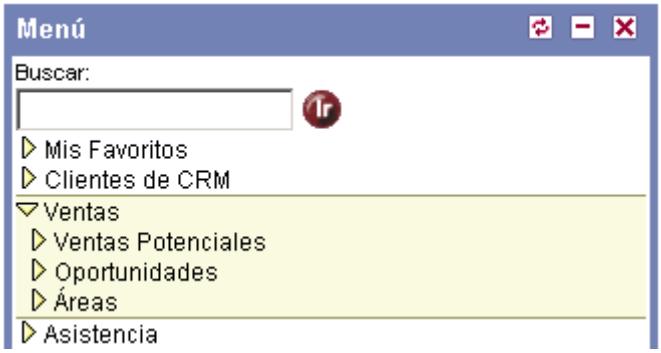
Si pinchamos en un enlace, navegaremos hasta la pantalla correspondiente a dicha opción.



Menú lateral

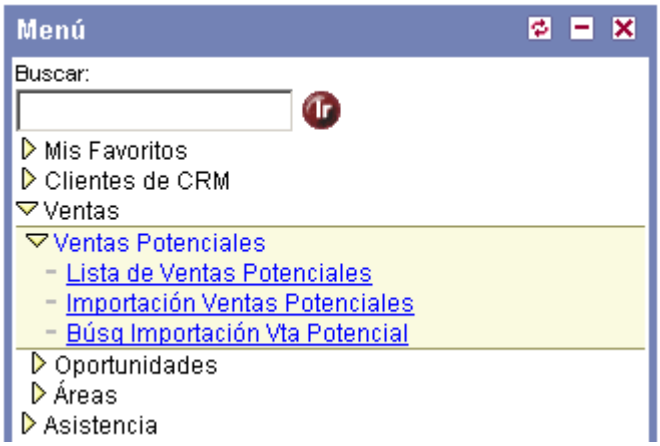
Por ejemplo, si pinchamos en “Ventas”, aparecerán los submenús de ventas.

Menú principal



Ventas- Submenús

Si repetimos la operación con la opción “Ventas potenciales”, aparecen los enlaces a las diferentes pantallas de la sección “Ventas potenciales”

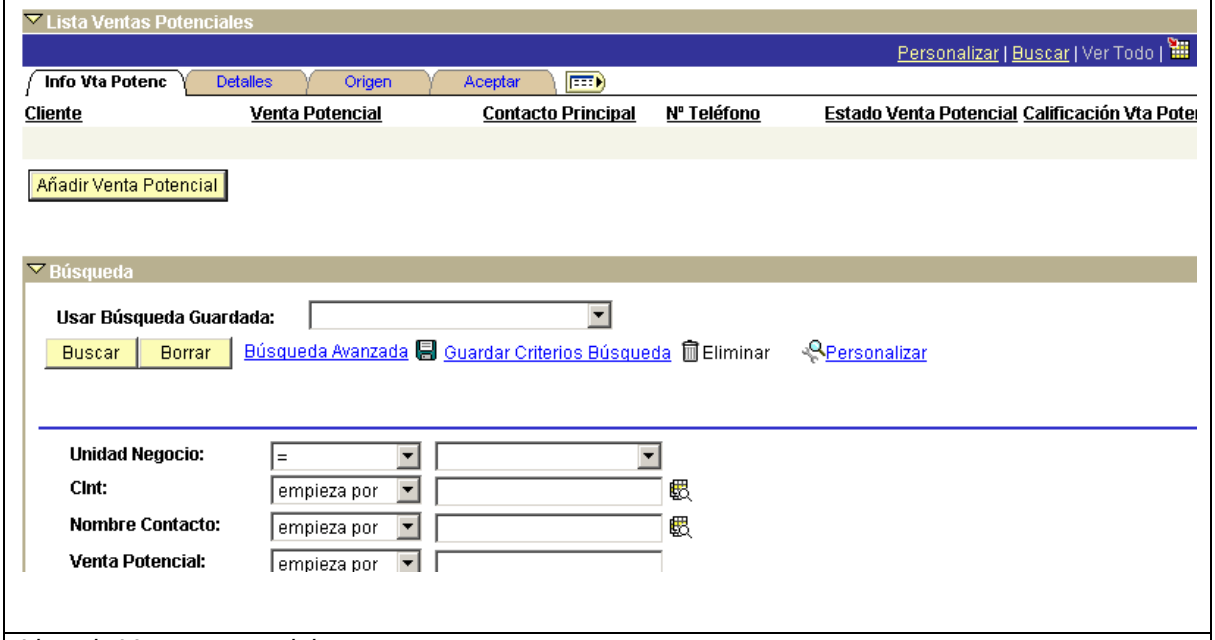


Ventas- Submenús

En cualquier momento podemos volver a plegar el menú, seleccionando una opción ya desplegada.

Si seleccionamos una opción distinta del menú, se plegará la opción previamente desplegada automáticamente.

Pinchando sobre el enlace “Lista de Ventas Potenciales”, accedemos a la pantalla correspondiente.



Lista de Ventas potenciales

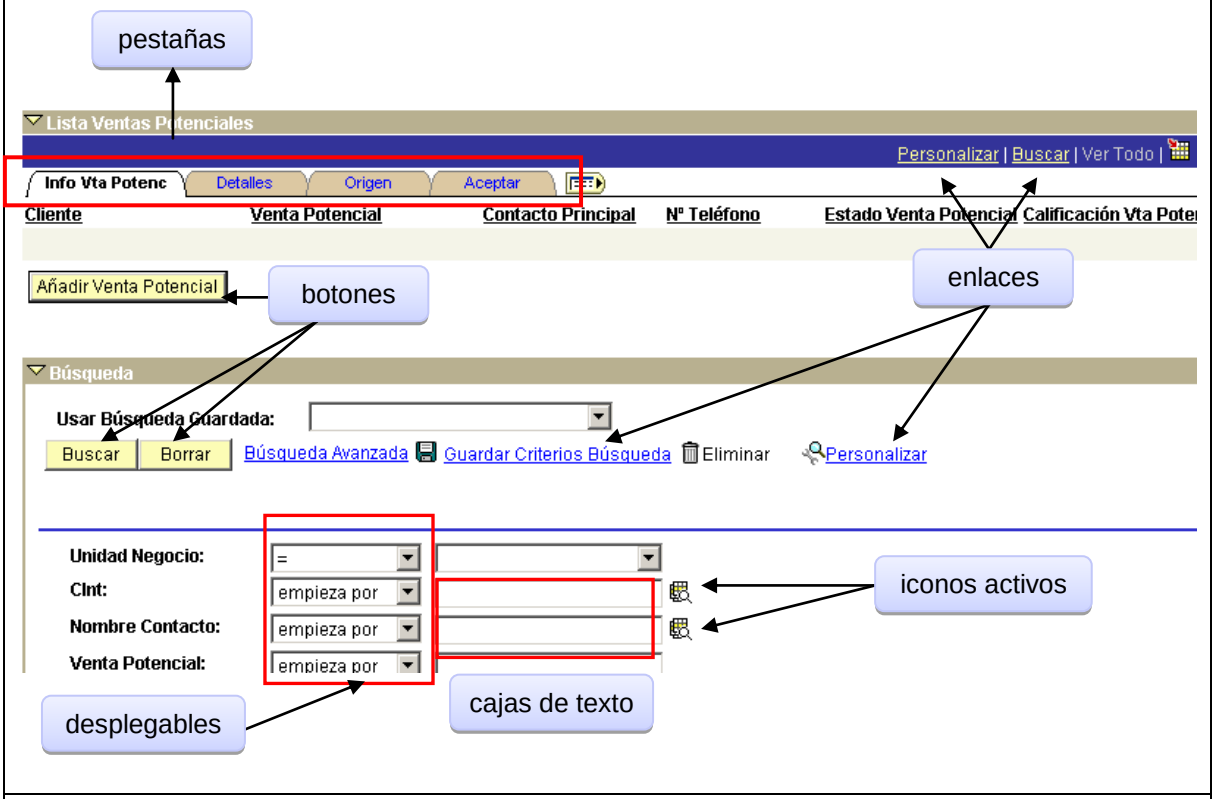
Utilizaremos esta pantalla para explicar otros elementos gráficos comunes que aparecen en la pantalla.

5.2.2 Otros elementos de navegación

Dentro de una pantalla del sistema de PeopleSoft podemos encontrar diferentes elementos gráficos que nos permiten realizar acciones. Las funciones que desempeñan estos elementos generalmente son estándar, por lo que su comportamiento es similar independientemente de la pantalla en la que nos encontremos. No todos los iconos sirven para realizar una acción. Cuando sobre un icono podemos realizar una acción, el puntero del ratón cambia al pasarlo por encima.

Entre estos elementos destacamos:

- Enlaces
- Pestañas
- Botones
- Cajas de texto
- Desplegables
- Iconos



Elementos de navegación

5.2.2.1 Enlaces

Los enlaces permiten navegar por al aplicativo haciendo transferencia entre distintas páginas así como activar diferentes funciones dentro del sistema como por ejemplo personalizar una página, abrir una ventana de búsqueda, etc.

5.2.2.2 Pestañas

Se utilizan para agrupar funciones dentro de una pantalla, de esta manera no es necesario cambiar de pantalla para trabajar con todos los datos.

Para cambiar de pestaña sólo es necesario pinchar en el nombre de la pestaña

Pestañas de navegación

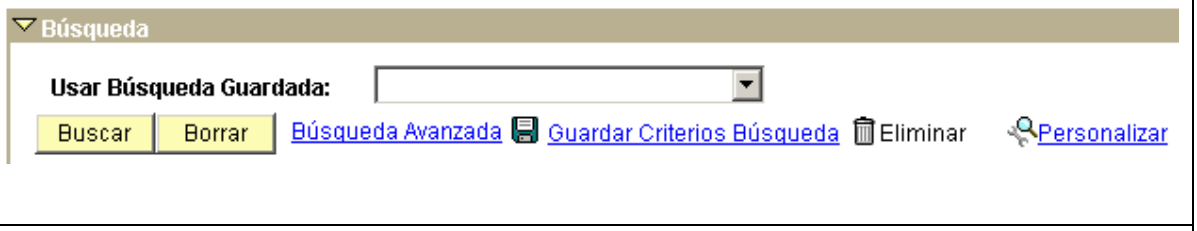
Por ejemplo, pinchamos en la pestaña de “Detalles”

Pestañas de navegación

5.2.2.3 Botones

Los botones se utilizan para desencadenar acciones. Los botones están etiquetados con la acción que realizan. Para desencadenar la acción del botón basta con situar el puntero encima de él y pulsar el botón izquierdo del ratón.

Por ejemplo, para empezar la búsqueda una vez escritos los criterios de búsqueda, pulsamos sobre el botón “Buscar”

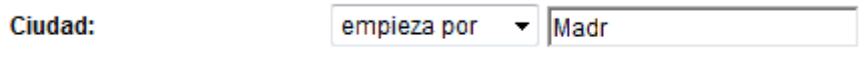


Botones

5.2.2.4 Cajas de texto

Las cajas de texto son una de las formas de introducir información en el sistema.

Por ejemplo, para realizar una búsqueda tenemos que introducir los criterios de búsqueda. Las cajas de texto permiten introducir el texto que el usuario escribe. Para ello pinchamos en el objeto caja, y luego escribimos



Caja de texto

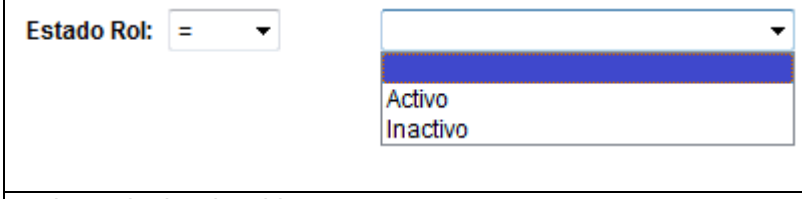
5.2.2.5 Desplegables

Los desplegables, son listas de valores que únicamente permiten seleccionar una opción de la lista, no permiten escribir nada en modo texto libre, ni añadir nuevos valores.

Por ejemplo, si nos fijamos en el campo “Estado Rol”, vemos que hay 2 desplegables. Cada uno de ellos tiene sus opciones, y sólo podemos seleccionar una opción en cada uno de ellos.

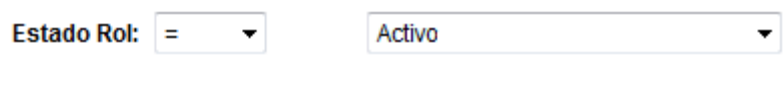
Desplegable

Para poder cambiar la selección pinchamos en cualquier punto del objeto, y aparece una lista con las posibles opciones seleccionables.



Valores de desplegable

Movemos el puntero por encima de las opciones, y seleccionamos aquella que nos interese. En el momento de pinchar sobre la opción, la lista de opciones se oculta.



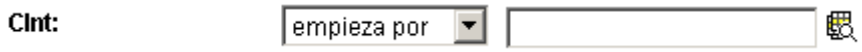
Selección de desplegable

Una vez seleccionada la opción nueva, esta aparece en el objeto.

5.2.2.6 Iconos

Los iconos son gráficos que pueden tener asociada funcionalidad. Esto significa que hay algunos iconos que pueden realizar algún tipo de acción al pinchar en ellos y otros que no realizan ninguna. Generalmente se pueden distinguir unos de otros al pasar el puntero por encima, si el icono del puntero lo señala, es que se puede realizar algún tipo de acción.

Por ejemplo en el caso del campo “Clnt”, vemos que hay un icono a la derecha de la caja de texto.



Icono

Si pinchamos en dicho icono, aparece una nueva pantalla de búsqueda

Pantalla abierta por icono

5.2.3 Esperas en el sistema

Cuando el sistema está realizando alguna tarea, una búsqueda, el almacenamiento de datos, o procesos, nos indica el estado en el que está en la parte superior derecha de la pantalla.

Estado "Procesando"

5.3 Acceso al sistema

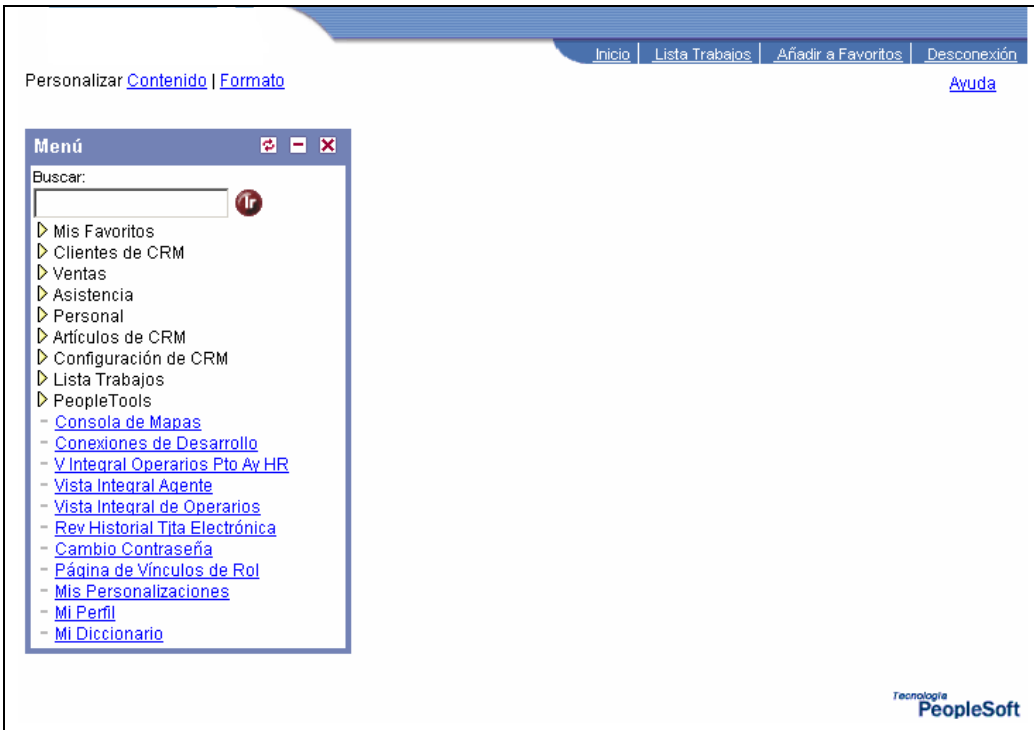
Para acceder al sistema, nos conectamos a la URL de la aplicación en un navegador web, y aparece una pantalla de control de acceso (login). Para poder entrar, debemos escribir nuestro nombre de usuario y la contraseña y pulsar después el botón de "Conexión".

Puesto que PeopleSoft es un aplicativo multilenguaje se da la posibilidad de logarse al mismo en distintos idiomas.

Control de acceso

Si hay un error de logado, el sistema nos lo indica. Por ejemplo, si la contraseña o el nombre usuario es incorrecto aparecerá un mensaje indicando que no son válidos.

Tras el logado aparecerá la siguiente pantalla:

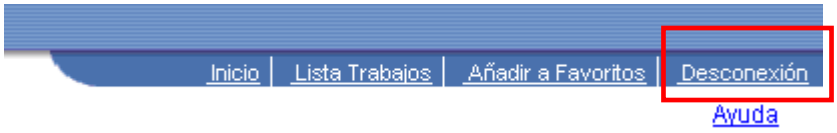


Pantalla principal

En la pantalla principal aparece un menú a la izquierda en el que podemos navegar a las diferentes secciones de la aplicación. Este menú es configurable según el perfil de acceso del usuario y permite realizar búsquedas para localizar la parte del menú a la que queramos acceder de forma sencilla.

5.4 Desconexión del sistema

Para salir de la aplicación deberemos desconectarnos. Para ello, utilizaremos el enlace(“Desconexión”) que aparece en la cabecera de todas las pantallas del sistema.



Desconexión del sistema

Es importante hacer una correcta desconexión del sistema de tal forma que no dejemos sesiones abiertas y se puedan optimizar los recursos, aunque también es posible configurar PeopleSoft para que se desconecte automáticamente tras un periodo de inactividad.

5.5 Funciones básicas del sistema

Dentro del sistema, existen funciones que aunque se apliquen a distintas páginas o componentes actúan de la misma manera facilitando las tareas.

Podemos distinguir las siguientes funciones:

- Realización de búsquedas
- Presentación de resultados: modo grid o páginas con pestañas.
- Acceso a los datos de una entidad
- Creación de datos de una entidad
- Modificación de datos de una entidad
- Exportación de resultados en distintos formatos: Excel, pdf.

Todas las funcionalidades de navegación y uso son aplicadas a las entidades básicas del CRM que hemos visto en el capítulo de “conceptos clave del CRM”. De esta forma se podrá por ejemplo:

- Crear, modificar entidades básicas explicadas: clientes, cuentas, servicios, sedes, personas de contacto, interacciones, casos, cuestionarios, proyectos de gestión.
- Creación o modificación de los atributos de las entidades: por ejemplo añadir un método de contacto de un cliente, cambiar la tipología de cliente, asociar un apoderado como persona legal de contacto de un cliente empresa, etc
- Consulta y extracción de datos de todas las entidades como de sus atributos correspondientes

5.5.1 Realización de búsquedas

Para acceder a los datos de las diferentes entidades dentro del sistema, generalmente se realiza una búsqueda primero para filtrar los resultados, y luego se escoge del conjunto de resultados aquellas entidades con las que se quiere trabajar.

Existen dos maneras de realizar búsquedas en el sistema:

- Utilizando un formulario para especificar los criterios de búsqueda.
- Utilizando una búsqueda almacenada

5.5.1.1 Formulario de búsqueda

Supongamos que queremos trabajar con ventas potenciales, para ello navegamos por el menú de la izquierda hasta la pantalla de ventas potenciales(Ventas-Ventas potenciales-Lista de ventas potenciales).

En esta pantalla encontramos un formulario para especificar los criterios de búsqueda. Dentro de los formularios de búsqueda encontramos dos tipos de elementos.

En este caso podemos realizar búsquedas por los siguientes conceptos:

- Unidad de Negocio
- Clnt (Cliente)
- Nombre contacto
- Venta potencial
- Estado venta potencial
- Calificación Vta Potencial

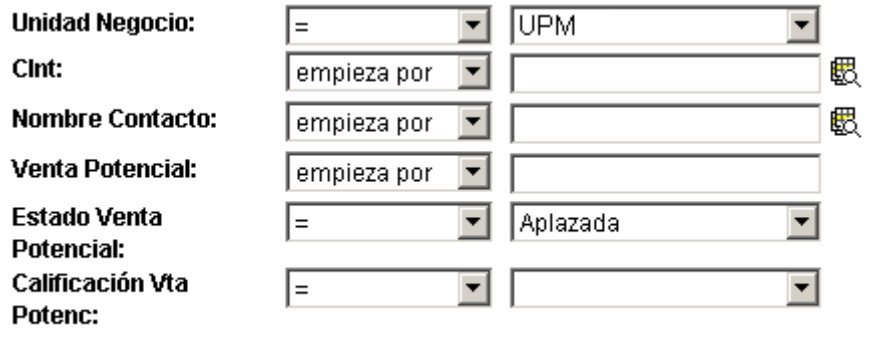
Por cada concepto de búsqueda tenemos que especificar dos cosas

- Qué valor queremos buscar
- Como queremos buscar ese valor

En este caso podemos utilizar los desplegables y las cajas de texto del formulario. Por ejemplo, supongamos que queremos buscar las oportunidades que pertenecen a la unidad de negocio “UPM” y que además su estado sea “Aplazada”

Dentro de los operadores de comparación tengo diferentes opciones:

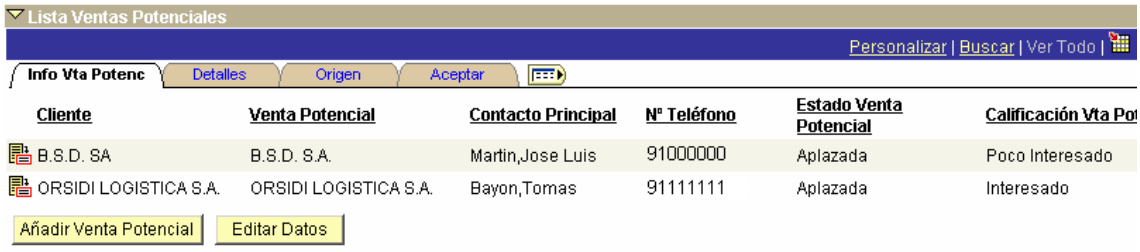
- Igual
- Vacío
- Empieza por
- Distinto de
- No incluido en



Búsqueda de datos

Si hay conceptos que no queremos utilizar en la búsqueda, simplemente no hay que asignar ningún valor en la columna de valores.

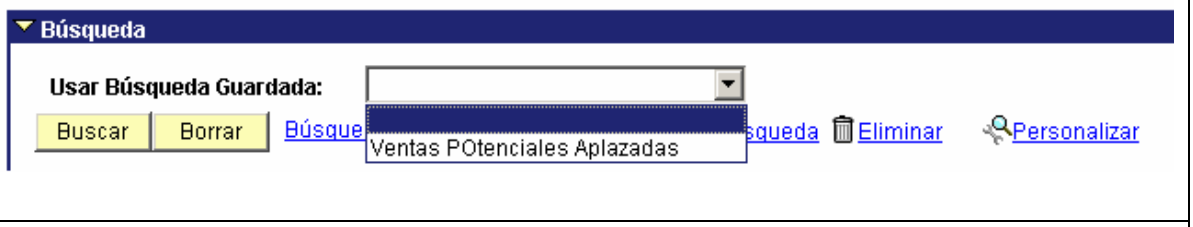
Una vez seleccionados los criterios, presionamos el botón “Buscar”, y si existen resultados que cumplan estos criterios, se mostrará una lista de resultados.



Lista de resultados

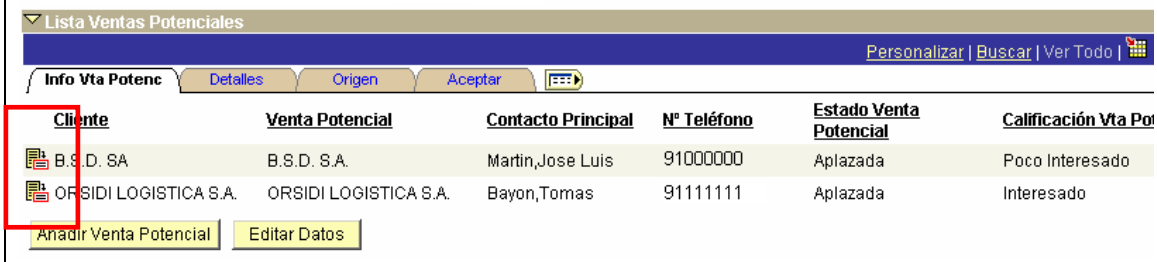
5.5.1.2 Búsquedas almacenadas

El sistema permite almacenar grupos de criterios de búsqueda y asignarles un nombre, de esta manera si una búsqueda se realiza con relativa frecuencia, en lugar de tener que volver a especificar todos los criterios cada vez, podemos escoger el nombre de la búsqueda y lanzarla.



Búsquedas guardadas

Por ejemplo en este caso en el desplegable existe una única búsqueda almacenada que contiene los criterios para localizar las ventas potenciales cuyo estado es “Aplazada”. Seleccionamos la búsqueda y pulsamos el botón “Buscar”.



Resultados de búsqueda guardada

Para acceder al detalle de una venta, y en general al detalle de cualquier entidad debemos pinchar en el icono que está a la izquierda del nombre.

5.5.2 Detalle de una entidad

Una vez realizada la búsqueda, para acceder a los datos de una entidad, seleccionamos de la lista de resultados.

Detalle de una venta potencial

6 MANUAL DE DESARROLLO

6.1 PeopleTools

PeopleTools es la plataforma de desarrollo de aplicaciones PeopleSoft. El principio fundamental que sigue y que ha sido parte de su éxito es que los desarrolladores y analistas deben centrarse en definir aplicaciones más bien que codificarlas íntegramente. Por ese motivo posee una extensa cantidad de objetos para la creación de aplicaciones sin necesidad de codificar completamente el aplicativo para su uso.

PeopleTools ofrece integración basada en estándares de mercado, tiene además informes robustos, características de interfaz de usuario avanzadas, herramientas administrativas y herramientas para la gestión de ciclo de vida de software y proyectos.

6.1.1 Integración

Para permitir la integración externa, PeopleSoft utiliza una plataforma de integración basada en estándares que se llama Integration Broker. Dicha plataforma cubre tres aspectos críticos para la integración:

- Definiciones de integración basadas en metadatos para un desarrollo y gestión de ciclo de vida sencillo.
- Plataforma de ejecución completa para el procesamiento síncrono y asíncrono.
- Adaptadores de tecnologías estándar como por ejemplo web services, FTP o correo electrónico.

Mediante Integration Broker, las aplicaciones PeopleSoft se pueden integrar con otras aplicaciones PeopleSoft o bien con sistemas externos utilizando tecnología de integración estándar como SOAP, XML, WSDL, WS_Security, FTP(s) o REST.

6.1.2 Informes y Analíticos

PeopleSoft provee herramientas web para localizar y organizar datos en tiempo real. Los usuarios pueden consultar datos concretos del sistema o datos operativos en tiempo real y acceder a dichos resultados desde diferentes dispositivos como un ordenador portátil o una Tablet.

Puesto que las tecnologías de reporte son parte core de PeopleTools no se requiere movimiento de datos para acceder a dichos datos en tiempo real. Usando la consola de reporte provista con PeopleTools, los usuarios pueden programar querys, informes y procesos que se ejecuten de forma recurrente distribuyendo los resultados mediante informes de forma segura.

Además permite embeber definiciones de reporte dentro de las aplicaciones de negocio para poder conseguir análisis de contexto.

6.1.3 Seguridad

PeopleSoft mantiene una política de seguridad basada en el modelo definido de persona, la interfaz de directorio y los roles dinámicos. Además, PeopleTools permite integrarse con otros productos de seguridad del propio Oracle para extender las capacidades de seguridad de PeopleSoft.

6.1.4 Gestión de ciclo de vida

PeopleTools tiene un conjunto completo de aplicaciones y utilidades de soporte de ciclo de vida. Las herramientas que provee son las siguientes:

- PEOPLESOFT UPDATE MANAGER: Herramienta para simplificar el proceso de aplicación de actualizaciones para conocer las necesidades del cliente.
- CHANGE ASSISTANT: Automatiza el proceso de instalación de parches.
- CHANGE IMPACT ANALYZER: Permite analizar el impacto de los cambios durante una actualización de la aplicación.
- SETUP MANAGER: Ayuda en la identificación, gestión y ejecución de las tareas de configuración.
- PEOPLESOFT TEST FRAMEWORK: Permite la creación de scripts de prueba automatizados.

6.1.5 Experiencia de usuario

PepleTools facilita un modelo de interacción sencillo basado en el uso de roles y dashboards como panel de control intuitivo. Facilita además la utilización de herramientas colaborativas como foros y blogs.

Soporta estándares HTML que permiten la rápida adopción de nuevas tecnologías. Esto hace que las interfaces de usuario de aplicaciones PeopleSoft sean adaptables y permite a los clientes alinear su experiencia de usuario con sus procesos de negocio.

6.1.6 Plataforma

PepleTools permite la elección de plataforma de tal forma que no se tenga que acometer una migración de la plataforma IT del cliente. Soporta todos los sistemas operativos líderes en la industria (AIX, HPUX, Linux, Solaris, Windows), productos RDMBS (Oracle, DB2, Informix, SQL Server, Sybase), servidores web (Welogic, Websphere) y de aplicaciones (Tuxedo, JRE) así como todos los navegadores web más populares (Internet Explorer, Firefox, Chrome, Safari).

6.1.7 Desarrollo

El Application Designer de PeopleTools es una plataforma madura y probada de desarrollo que permite ciclos de vida de desarrollo rápido para funcionalidad nueva y personalizada. Permite la creación de funcionalidad y aplicaciones PeopleSoft de forma rápida

La potencia de Application Designer hace de PeopleTools una ventaja estratégica para muchos clientes.

6.2 PeopleCode

PeopleCode es el lenguaje propietario para el desarrollo de aplicaciones PeopleSoft. Es un lenguaje de programación estructurado y se utiliza tanto en la programación de componentes del online como en la programación de procesos batch.

Es un lenguaje interpretado, aunque en la programación de procesos batch existe la posibilidad de compilar el código.

En este proyecto fin de carrera vamos a ver los siguientes aspectos fundamentales sobre el lenguaje PeopleCode que nos va a permitir programar aplicaciones PeopleSoft:

- Tipos de Datos
- Sentencias
- Funciones
- Expresiones
- Variables
- Operadores

6.2.1 Tipos de Datos

En las definiciones de tipos de datos de PeopleCode podemos encontrarnos dos tipos de datos claramente diferenciados:

- **Tipos de dato convencionales:** Componen el núcleo de la funcionalidad de PeopleCode.
- **Tipos de dato de objeto:** Usados para instanciar objetos de las clases propias de PeopleSoft.

6.2.1.1 Tipos de dato convencionales

Los tipos de dato convencionales son los propios que podríamos encontrarnos en cualquiera de los lenguajes de programación más utilizados.

Los tipos de datos convencionales usados por PeopleCode son los siguientes:

- ANY: Es un tipo de dato indeterminado que es calculado por el intérprete en tiempo de ejecución. Habitualmente se utiliza en la devolución de valores.
- BOOLEAN: Para valores boléanos.
- DATE: Para valores de tipo fecha.
- DATETIME: Para valores de tipo fecha y hora.
- TIME: Para valores de hora.
- FLOAT: Para valores reales con notación de coma flotante.
- INTEGER: Para valores enteros de 32 bits con signo con el rango -2.147.483.648 a 2.147.483.647

- NUMBER: Para valores numéricos.
- STRING: Para cadenas de texto.
- OBJECT: Para objetos.

6.2.1.2 Tipos de dato de objeto

Son tipos de datos especiales utilizados para trabajar con los objetos propios de PeopleSoft.

Algunos de los tipos de dato de objeto más importantes y utilizados son:

- Field: Para campos de registros
- Record: Para registros
- Row: Para acceso como buffer de datos
- Rowset: Para acceso como buffer de datos
- Grid: Para elementos de páginas
- Page: Para paginas
- Array: Para cadenas de elementos
- File: Para ficheros
- SQL: Para acceso a base de datos
- ApiObjet: Para cualquier objeto de API (objetos de sesión, component interfaces…)

6.2.2 Comentarios

Se pueden utilizar 3 tipos de comentarios distintos dentro del código de PeopleCode:

- Opción 1: Empezando con /* y finalizando con */
- Opción 2: Utilizando la expresión REM y acabando el comentario con ;
- Opción 3: Empezando con <* y finalizando con *> para anidar comentarios.

Un ejemplo de los distintos comentarios que podríamos encontrar dentro de un código PeopleCode podría ser el siguiente:

```
<* Este es un bloque entero de código comentado
rem comentario de PeopleCode;

/* ----- Llamada a dos funciones ----- */

/* Llamada a Funcion Principal  */
```

```
Llamada_a_funcion(&Param1, 1, 0);

/* Llamada a Funcion Secundaria  */
Llamada_a_funcion_sec(&Param2);

*>
```

6.2.3 Sentencias

La sentencia es el elemento básico en el que se divide el código de los programas PeopleCode. Puede ser por ejemplo una declaración, una asignación, un constructor de programa o una llamada a una función.

En referencia a las sentencias vamos a ver los siguientes aspectos:

- Separadores
- Sentencias de asignación
- Constructores del lenguaje
- Sentencia de control
- Bucles condicionales

6.2.3.1 Separadores

El separador para concluir sentencias en PeopleCode es el punto y coma (;). Este separador es admitido incluso cuando no es requerido (por ejemplo al final de una sentencia if no es requerido ningún separador pero también se admite el separador punto y coma para concluir).

Los espacios en blanco extras y los saltos de línea son ignorados por el intérprete.

6.2.3.2 Sentencias de asignación

Es la sentencia básica de PeopleCode y consta de una variable a la izquierda, un operador de igualdad y una expresión a la derecha.

```
nombre = expresion
```

Existen sentencias de asignación por valor (asignación directa) o por referencia (almacenando la dirección de memoria donde se encuentra el dato).

Un ejemplo de asignación:

```
Local array of number &arrNumeros, &arrNumeros2;
Local number &nbrNumero;


/* Asignacion por valor */
&nbrNumero = 1;

/* Asignación por referencia*/
&arrNumeros = CreateArray(2, 4, 8);
&arrNumeros2 = &arrNumeros;
```

6.2.3.3 Constructores del lenguaje

Los constructores del lenguaje PeopleCode son los siguientes:

- Estructuras de bifurcación: **If** y **Evaluate**.
- Bucles: **For, Repeat,** y **While.**
- Constructores de escape de bucles: **Break** y **Exit.**
- Retorno de funciones: **Return**.
- Sentencias de declaración de variables y funciones: **Global, Component, Local** y **Declare Function.**
- Definición de funciones: **Function**.
- Sentencia de definición de clases: **Class**.
- Sentencias de manejo de errores: **Try, Catch** y **Throw**.

6.2.3.4 Sentencias de control

Son sentencias para controlar el flujo de ejecución dentro de los programas con PeopleCode.

- **If...Then...Else**

Sintaxis:

```
If condicion Then
   [lista_de_sentencias_1]
[Else
   [lista_de_sentencias_2]]
End-if;
```

Evalúa la condición, si es cierta ejecuta la lista_de_sentencias_1 y en otro caso la lista_de_sentencias_2.

La cláusula Else es optativa y puede aparecer o no dependiendo de la utilidad con la sentencia If.

Ejemplo:

```
If &nbrNumero1 > &nbrNumero2 Then
   WinMessage ("El numero " | &nbrNumero1 | " es mayor que " | &nbrNumero2);
Else
   WinMessage ("El numero " | &nbrNumero2 | " es mayor que " | &nbrNumero1);
End-if;
```

- **Evaluate**

Sintaxis:

```
Evaluate term_izq
   When [operador_relacional_1] term_dcha_1
     [lista_de_sentencias]
     [Break;]
   .
   .
   .
   When [operador_relacional_n] term_dcha_n
     [lista_de_sentencias]
     [Break;]
   [When-other
      [lista_de_sentencias]]
End-evaluate;
```

Toma el valor de term_izq y lo evalúa en base al operador relacional comparándolo con term_dcha. En caso de que la comparación sea cierta ejecuta la lista de sentencias.

Si se utiliza la cláusula Break, la evaluación terminará al ejecutarse esta sentencia si no continuará evaluando la sentencia.

Ejemplo:

```
evaluate &strFrecuenciaUso
when = "nunca"
   &nbrFrecuencia = 0;
   Break;
when = "aveces"
   &nbrFrecuencia = 1;
   Break;
when = "frecuentemente"
   &nbrFrecuencia = 2;
   Break;
when-other
```

```
   Error "Valor no valido"
end-evaluate;
```

- **For**

Sintaxis:

```
For contador = expresion1 to expresion2
    [Step i];
    lista_de_sentencias
End-for;
```

Repite una secuencia de sentencias un número determinado de veces hasta que se cumpla el valor fijado para la iteración incrementando el contador i veces en cada iteración. Si se omite la cláusula Step i el valor de iteración será 1.

Ejemplo:

```
&nbrMax = 10;

for &nbrCount = 1 to &nbrMax;
   WinMessage("El valor del contador es  " | &nbrCount);
end-for;
```

6.2.3.5 Bucles condicionales

Los bucles condicionales repiten una secuencia de sentencias evaluando una expresión condicional cada vez que se ejecuta el bucle. El bucle concluye cuando la expresión evaluada es cierta. En PeopleCode existen dos bucles condicionales: Repeat y While

- **Repeat:**

Sintaxis:

```
Repeat
  lista_de_sentencias
Until expresion_logica;
```

Repite la lista de sentencias hasta que la condición de la sentencia Until se cumple como cierta.

Ejemplo:

```
&nbrNumero = 1
```

```
Repeat
  WinMessage("El valor del número es  " | &nbrNumero);
  &nbrNumero = &nbrNumero + 1;
Until &nbrNumero = 10;
```

- **While**

Sintaxis:

```
While expression_logica
     lista_de_sentecias
End-while;
```

Mientras la expresión lógica de la cláusula While sea cierta ejecuta la lista de sentencias.

Ejemplo:

```
&nbrNumero = 1

While &nbrNumero < 10
  WinMessage("El valor del número es  " | &nbrNumero);
  &nbrNumero = &nbrNumero + 1;
End-While;
```

6.2.4 Funciones

En PeopleCode podemos encontrarnos cuatro tipologías de funciones claramente diferenciadas:

- **Built-in:** Son las funciones estándar provistas por PeopleSoft para el desarrollo de aplicaciones con PeopleCode.
- **Internas:** Funciones definidas dentro del PeopleCode con la cláusula Function
- **Externas PeopleCode:** Definidas en PeopleCode pero fuera del propio programa, habitualmente en registros como librerías de funciones.
- **Externas No PeopleCode:** Almacenadas en librerías externas y con código que no es PeopleCode.

6.2.4.1 Definir funciones

Se pueden definir funciones de PeopleCode en cualquier programa, pero deben ir siempre definidas al principio del código.

Sintaxis:

```
Function nombre [(lista_de_parametros)] [Returns tipo_de_dato]
   [sentencias]
End-function

donde

lista_de_paremetros es: &param1 [As data_type] [, &param2 [As data_type]]...

tipo_de_dato es: Number, String, Date, Time, Datetime, Boolean, Object, o Any

sentencias es: una lista de sentencias PeopleCode
```

La definición de la función consta de las siguientes partes:

- La palabra clave **Function** seguida del nombre de la funciona y la lista de parámetros opcionales.
- La cláusula opcional **Returns** que especifica el tipo de dato que devolverá la función.
- Las sentencias que se ejecutarán cuando se llame a la función.
- La palabra clave **End-function** para finalizar la definición de la función.

Ejemplo:

```
function MayorQue(&nbrNumero1, &nbrNumero2) returns Boolean;

   If &nbrNumero1 > &nbrNumero2 Then
      Return True;
   Else
      Return False;
   End-if;

End-function;
```

Por definición, las funciones de PeopleCode se almacenan en registros cuyo nombre empieza por FUNCLIB_ y se guardan siempre en el campo deseado dentro del evento FieldFormula.

Si la función se utiliza dentro del ámbito del programa en el que se ha definido, no necesita ser declarada. Si se quiere llamar a una función externa al ámbito del programa se debe declarar la función al principio del código.

6.2.4.2 Declarar funciones

Es necesario declarar la función siempre que se desee llamar a una función definida fuera del ámbito del programa.

Existen dos maneras de declarar las funciones dependiendo de si se trata de funciones programadas en PeopleCode o si son funciones compiladas en una biblioteca de enlace dinámico (DLL).

6.2.4.2.1 Funciones PeopleCode

Sintaxis:

```
Declare Function nombre_funcion PeopleCode nombre_registro.nombre_campo
tipo_evento
```

La declaración de la función consta de las siguientes partes:

- Las palabras clave **Declare Function** seguida del nombre de la función.
- La palabra clave **PeopleCode** que indica que es una función con código de tipo PeopleCode.
- nombre_registo.nombre_campo para indicar el registro y el campo donde está almacenada la función.
- tipo_evento para indicar el evento asociado a la función.

Ejemplo:

```
declare function validarNIF PeopleCode FUNCLIB_UTILES.DOCUMENTOS FieldFormula;
```

6.2.4.2.2 Funciones externas (compiladas en DLL)

Sintaxis

```
Declare Function nombre_funcion Library nombre_libreria
   [ALIAS nombre_funcion_modulo ]
   [lista_de_parametros]
   [RETURNS ext_return_tipo [As pc_tipo]]

Donde:
lista_de_parametros:   ([ext_param1 [, ext_param2] ...)
ext_paramn:   ext_tipo_de_dato [{REF|VALOR}] [As pc_return_tipo]
```

La declaración de la función consta de las siguientes partes:

- Las palabras clave **Declare Function** seguida del nombre de la función.
- La palabra clave **Library** que indica que es una función con código externo seguido del nombre de la librería.
- De forma opcional la palabra clave **ALIAS** seguida del nombre de la función en modulo en el caso de que este difiera del nombre de la función declarada en PeopleCode.
- lista_de_parametros para indicar los parámetros necesarios para la ejecución de la función con la forma: e*xt_tipo_de_dato* [{**Ref**|**VAlor**}] [**As** *pc_return_type*]
    - ext_tipo_de_dato: Tipo de dato del parámetro (BOOLEAN, INTEGER, LONG, UINTEGER, ULONG, STRING, FLOAT, DOUBLE)
    - [{**Ref**|**VAlor**}]: Indica si el parámetro es pasado por referencia o por valor.
    - **As** *pc_return_tipo:* Indica el tipo de dato PeopleCode (String, Number, Date, Boolean o Any)
- De forma opcional la palabra clave **RETURNS** seguido del tipo de dato devuelto por la función (BOOLEAN, INTEGER, LONG, UINTEGER, ULONG, FLOAT, DOUBLE) y la palabra clave **As** para indicar el tipo de dato de la variable de PeopleCode que recogerá el valor devuelto por la función (String, Number, Date, Boolean o Any)

Ejemplo:

```
declare function validarNIF library "DOCUMENTOS.dll" (integer value as number)
returns boolean as boolean;
```

6.2.4.3 Llamar a funciones

Una vez definida y en caso de ser necesario declarada, podemos proceder a llamar a la función. La llamada de funciones sigue la siguiente sintaxis:

```
Nombre_de_la_funcion([lista_de_parametros])
```

La llamada a la función se realiza con el nombre de la función definido previamente y de manera opcional la lista de parámetros definidos en la función. Si se desea almacenar el resultado del valor devuelto por la función entonces habrá que llamar a la función asignando el valor de retorno en una variable para su utilización.

```
&bResultado = validarNIF(&nbrNIF);
```

En las funciones internas o externas de PeopleCode los parámetros se pasan siempre por referencia. En las funciones del tipo built-in dependerá de la definición de la propia función si se pasa el parámetro por valor o por referencia. En las funciones externas no PeopleCode también dependerá de la propia declaración de la función.

6.2.4.4 Valores de retorno de funciones

Las funciones pueden devolver valores de los tipos de datos soportados por PeopleCode. Algunas funciones incluso pueden no devolver ningún tipo de valor, pero esto no es inusual y únicamente será para el caso de algunas funciones predefinidas built-in. El resto de funciones definidas por el programador deberán devolver siempre algún valor.

6.2.5 Expresiones

Las expresiones evalúan valores de cualquiera de los tipos de datos soportador por PeopleCode. Una expresión simple de PeopleCode puede ser una constante, una variable temporal, una variable de sistema, una referencia a un campo de registro o una llamada a una función. Las expresiones simples pueden ser modificadas por operadores (por ejemplo el signo negativo o el lógico NOT). También pueden ser combinadas usando operadores binarios (por ejemplo el signo de suma o el AND lógico).

Las referencias de definición de nombre (Definition Name References) se utilizan para referenciar objetos de PeopleTools como puede ser registros, páginas o mensajes.

Existen en PeopleCode unas expresiones especiales llamadas Metastrings o también meta-SQL. Estas expresiones se utilizan dentro del código SQL y en tiempo de ejecución se convierten en propio código SQL correspondiente a la plataforma de Base de Datos, de esta

forma con los Metastring somos capaces de aislarnos de las particularidades propias de la BBDD.

Las expresiones utilizas por PeopleCode que veremos son:

- Constantes
- Funciones usadas como expresiones
- Variables
- Variables de sistema.
- Metastrings o meta-SQL.
- Referencias de campos de registro.
- Referencias de definición de nombre (Definition Name Referentes)

6.2.5.1 Constantes

Las constantes soportadas por PeopleCode son de tipo numérico, boolenanas y constantes definidas por el propio usuario.

Para expresar valores del tipo Date, DateTime y Time se utilizan funciones predefinidas (built-in) de PeopleCode que convierten de tipo String a constantes numéricas (funciones Date, Date3, DateTime6, DateTimeValue, DateValue, Time3, TimePart, y TimeValue), así como también se puede hacer la conversión inversa de un valor Datetime a texto utilizando la función FormatDateTime.

- Constantes numéricas: Cualquier número decimal (por ejemplo: 7, 3.1416, -142).
- Constantes tipo string: Delimitadas por la comilla simple (‘) o doble comilla (‘’).
- Constantes booleanas: Valor verdadero (True) o Falso (False).
- Constante Null: Representa un valor de objeto que no se referencia a un objeto válido.
- Constantes definidas por el usuario: Se definen al principio del programa de modo local (no se permiten constantes globales en PeopleCode) con la palabra clave *Constant*.

```
Constant &bValorCierto = True;
Constant &nbrNumeroSiete = 7;
Constant &strNombre = "Daniel";
```

6.2.5.2 Funciones usadas como expresiones

Un tipo de expresión común que se puede utilizar es una función que devuelva un determinado valor. Se utilizan en el lado derecho de una sentencia de asignación, como parámetro a otra función o en combinación con otras expresiones para formar una expresión compuesta.

```
&bResultado = validarNIF(&nbrNIF);
```

6.2.5.3 Variables

Dedicaremos un apartado posterior (*6.2.6 Variables*) de forma íntegra para explicar las variables debido a su importancia y la necesidad de una explicación más extensa.

6.2.5.4 Variables de sistema

Las variables de sistema se identifican en PeopleCode por ir precedidas por el símbolo **%** en las partes del código donde aparecen. Se utilizan por ejemplo para obtener la fecha actual, información de usuario, idioma, etc.

```
&dtFecha = %Datetime;
```

6.2.5.5 Metastrings o meta-SQL

Los Metastrings, también conocidos como meta-SQL, son sentencias especiales PeopleCode de tipo SQL. Van precedidas por el símbolo **%** al igual que las variables de sistema. La peculiaridad que tiene este tipo de expresiones es que en tiempo de ejecución tienen la capacidad de traducirse al lenguaje de la plataforma de BBDD. Es una forma especial utilizada por PeopleCode para la escritura de sentencias SQL que independiza la misma de la BBDD.

```
SQLExec("%SelectByKeyEffDt(:1,:2)", &DST, %Date, &DST);
```

6.2.5.6 Referencias de campos de registro.

Las referencias de campo de registro son expresiones que se utilizan tanto para recuperar valores almacenados en campos de registro como para asignar valores a dichos campos.

La sintaxis que se utiliza:

```
[recordname.]fieldname
```

Ejemplo de uso:

```
&REC1.NOMBRE.Value = "Daniel";
&strNombre = &REC1.NOMBRE.Value;
```

6.2.5.7 Referencias de definición de nombre (Definition Name Referentes)

Las referencias de definición de nombre, conocidas en PeopleCode como Definition Name References, son un tipo de expresiones especiales que hacen alusión a definiciones de nombres de objetos específicos de PeopleTools, como por ejemplo registros, páginas, componentes, etc. Sintácticamente, una referencia de definición de nombre consiste en una palabra reservada indicando el tipo de definición seguido por un punto y el nombre de la definición de PeopleTools.

```
Local Record &REC1;
&REC1 = CreateRecord(Record.CLIENTE);
```

Las referencias de definición de nombre que hay en PeopleTools son las siguientes:

Referencia de def de nombre	Uso
BARNAME	Utilizada para gestión de páginas.
BUSACTIVITY	Utilizada con la clase “TriggerBusinessEvent”.
BUSEVENT	Utilizada con la clase “TriggerBusinessEvent”.
BUSPROCESS	Utilizada con la clase “TriggerBusinessEvent”.

COMPINTFC	Utilizada con la clase "Component Interface".
COMPONENT	Utilizada para gestión de páginas y para generación de URLs.
FIELD	Utilizada para designar campos de registros.
FILELAYOUT	Utilizada con la clase "SetFileLayout File".
HTML	Utilizada con la función GetHTMLText.
IMAGE	Utilizada con los métodos y funciones para referenciar imágenes.
INTERLINK	Utilizada con la función "GetInterlink"
ITEMNAME	Utilizada para gestión de páginas.
MENUNAME	Utilizada para gestión de páginas.
MESSAGE	Utilizada con las funciones y métodos de mensajes.
MOBILEPAGE	Utilizada para la identificación de páginas.
NODE	Utilizada para gestión de páginas y para generación de URLs.
PAGE	Utilizada para gestión de página para pasar el nombre de la página
PORTAL	Utilizada para gestión de páginas y para generación de URLs.
RECORD	Utilizada en funciones y métodos para designar un registro.
SCROLL	Utilizada para obtener el nombre del área de scroll de la página.
SQL	Utilizada con definiciones SQL.
STYLESHEET	Utilizada con hojas de estilo.

6.2.6 Variables

En este apartado vamos a analizar el uso de las variables con PeopleCode. Existen dos tipos distintos de variables a utilizar dentro de los programas de PeopleCode:

- Variables de sistema: Son variables que proveen información de sistema. Van precedidas del carácter '**%**'. Algunas de las variables más utilizas son para obtener la fecha, la hora, el usuario logado, la página actual, el componente, etc.
- Variables definidas por el usuario: Definidas por el usuario con el carácter **'&**'. La longitud del nombre es hasta 1000 caracteres alfanuméricos incluidos los caracteres #, @, $, y _

6.2.6.1 Declaración de variables de usuario y alcance

Según el periodo de vida de las variables podemos tener los siguientes tipos:

- Global: Son variables globales a toda la sesión.
- Component: Son variables válidas mientras que la página o el componente en el que están definidas está activo.
- Local: Son variables válidas durante la vida del programa PeopleCode o la función en las que han sido definidas.

Se pueden usar las variables declarándolas previamente con las palabras clave Global, Component o Local o bien se pueden utilizar sin una declaración previa.

```
Global String &strNombre;
Component RowSet &rstRow;
Local Number &nbrNumero;
Local Any &Valor;
Local Boolean &bSeguir = True;
```

Habitualmente las variables se declaran al inicio del programa de PeopleCode, menos en el caso de las variables locales declaradas dentro de funciones o en la sección principal de un programa. Se declaran con cualquiera de los tipos válidos de PeopleCode. Existe la posibilidad de declararlas con el tipo de dato Any o simplemente no declararlas, en ese caso, en tiempo de ejecución se le asignará el tipo adecuado basándose en el contexto. Aun así,

siempre es recomendable declarar las variables para que se les asigne el tipo adecuado y que no haya errores en la asignación de tipos.

Las variables globales mantienen su valor mientras dure la sesión y pueden ser utilizadas por diferentes componentes y aplicaciones, incluidos los programas Application Engine. Las variables globales necesitan siempre ser declaradas, aunque se desaconseja su uso debido a la complejidad que conlleva su mantenimiento.

Las variables de tipo Component representan el punto medio entre las variables globales y las locales. Están definidas y mantienen sus valores mientras que cualquier página del componente en el que están definidas se mantenga activa. Al igual que las variables globales, deben ser declaradas en programa PeopleCode donde se quieran utilizar.

Las variables locales declaradas fuera de una función en un programa, están definidas y mantienen sus valores a lo largo de la vida de todo el programa. Las variables locales declaradas dentro de una función solamente mantienen su valor dentro del ámbito de la función en la que están definidas.

6.2.6.2 Declarando variables

Es recomendable declarar siempre todas las variables. Sin embargo, si una variable no se declara, podrá ser utilizada y PeopleSoft le asignará automáticamente el ámbito local y el tipo de dato Any. Al guardar el programa PeopleCode, se recibirán alertas de tipo warning indicando las variables que no han sido declaradas.

Aun así, el declarar las variables con el tipo de dato correcto evitará que se tengan errores en tiempos de ejecución por tipos erróneos o a la hora de guardar el programa por una incorrecta asignación.

6.2.6.3 Inicializando variables definidas por el usuario en la declaración

Las variables de tipo local permiten su inicialización en una única sentencia en el momento de la declaración de las mismas.

```
Local String &strNombre = "Daniel";
Local Number &nbrNumero = 3;
Local Boolean &bSeguir = True;
Local Date &dtFecha = %Date;
```

Por defecto las variables de tipo string y fechas están inicializadas a null y los números a cero.

6.2.6.4 Restricciones del uso de variables

Existen algunas restricciones que hay que tener en cuenta en el uso de los siguientes tipos de datos según el ámbito de declaración:

- Tipos de datos específicos solamente para variables locales (no globales ni componente):
  - JavaObject
  - Interlink (global solo para programas App Engine)
  - TransformData
  - XmlNode
- Los tipos de dato ApiObject deben ser locales excepto los siguiente que deben ser globales:
  - Session
  - Colección PSMessages
  - PSMessage
  - Todas las clases de tipo Tree (árboles, estructuras de árboles, nodos, niveles, etc.)
  - Clases Query

6.2.6.5 Consideraciones sobre el uso de variables locales

Una variable local que se declara en la parte principal de un programa, es decir, aquella que no está declarada dentro de una función, se puede ver afectada por una sentencia en cualquier parte del programa.

Pongamos un ejemplo útil: Supongamos que tenemos un programa PeopleCode en el que tenemos dos funciones (Funcion1 y Funcion2) declaradas en el campo RECORD_1.CAMPO_1 FieldFormula y ambas funciones utilizan la variable local &Temp. Las dos funciones podrán modificar dicha variable ya que están dentro del mismo programa PeopleCode. Sin embargo si ahora definimos otra función (Funcion3) en el campo

RECORD_2.CAMPO_2 FieldFormula y se hace referencia una variable llamada &Temp, esta última variable &Temp será otra distinta. Esto es especialmente importante si la función Funcion1 llamara a Funcion3. Técnicamente podríamos decir que ambas funciones existen al mismo tiempo, una dentro de otra, pero la variable &Temp es diferente para cada función.

6.2.6.6 Tiempo de vida de las variables locales

Como ya se ha comentado, una variable local es válida durante el periodo de vida de un programa PeopleCode o en la función donde se ha declarado. El matiz importante es establecer qué es el periodo de vida de un programa PeopleCode.

Cuando el sistema está evaluando un programa PeopleCode y se llama a una función en el mismo programa, no se inicia una nueva evaluación de programa. Sin embargo, cuando se llama a una función de un programa PeopleCode diferente, el programa actual se suspende y se inicia la evaluación del programa nuevo. Eso significa que cualquier variable local del programa llamante no estará disponible en el llamado. Aunque las variables locales se llamaran igual (con el mismo nombre) en ambos programas, se consideran variables distintas almacenadas de forma separada.

6.2.6.7 Pasando variables a funciones

Las variables que se utilizan en PeopleCode se pasan siempre a las funciones por referencia. Eso significa, entre otras cosas, que la función puede cambiar el valor de la variable y ser devueltas con el valor modificado.

```
Local String &strNombreCompleto = "Daniel Ballesteros Navarro";
Local String &strResultado;

[…]

/* Llamada a una función para quitar espacios */
QuitarEspacios (&strNombreCompleto);
&strResultado =  "El nombre sin espacios es: " | &strNombreCompleto;
```

6.2.6.8 Funciones recursivas

PeopleCode permite el uso de funciones recursivas. Una función puede llamarse a sí misma y cada posible llamada recursiva a la función tendrá su propia copia independiente de parámetros y variables locales de la función. Cuando se escribe funciones recursivas hay que

ser cuidadoso a la hora de pasar variables como parámetros debido a que PeopleCode hace la llamada por referencia.

6.2.7 Operadores

Las expresiones de PeopleCode se pueden modificar utilizando distintos operadores. Los operadores que vamos a ver son los siguientes:

- Operadores matemáticos
- Operadores para fecha y hora (date/time)
- Concatenación de strings
- El operador especial @
- Operadores de comparación
- Operadores booleanos

6.2.7.1 Operadores matemáticos

En PeopleCode se utilizan los siguientes operadores matemáticos:

Operador	Significado
+	Suma
-	Resta o signo negativo
*	Multiplicación
/	División
**	Exponencial

El orden de precedencia que se sigue es el de expresiones matemáticas evaluadas de izquierda a derecha. El operador exponencial se evalúa antes que la multiplicación y división y la multiplicación y división antes que la suma y resta.

6.2.7.2 Operadores para fecha y hora (date / time)

Es posible sumar o restar valores de tipo fecha/hora (date/time) obteniéndose como resultado un valor numérico.

Hay que recordar que en PeopleCode para manejo de fechas teníamos los siguientes tipos de datos:

- DATE: Para valores de tipo fecha.
- DATETIME: Para valores de tipo fecha y hora.
- TIME: Para valores de hora.

En el caso de operaciones de suma/resta sobre este tipo de datos, el valor numérico devuelto para datos tipo fecha representa la diferencia en días entre las dos fechas. Para datos tipo hora, representa la diferencia en segundos. También es posible sumar y restar valores numéricos a valores fecha/hora obteniéndose como resultado un valor fecha/hora.

El resumen de estos tipos de operaciones sería el siguiente:

Operación	Tipo de resultado	Significado
Hora + número de segundos	Dato tipo Time	Hora resultante de sumar
Fecha + número de días	Dato tipo Date	Fecha resultante de sumar
Fecha - Fecha	Dato tipo Number	Diferencia en días
Hora - Hora	Dato tipo Number	Diferencia en segundos
Fecha + Hora	Dato tipo DateTime	Fecha y hora combinados

6.2.7.3 Concatenación de strings

El operador utilizado para la concatenación de strings es el pipe (|). Este operador tiene la particularidad de que automáticamente convierte a los operandos en datos de tipo string.

```
Local String &strNombre = "Daniel";
Local Date &dtHoy= %Date;

WinMessage("Mi nombre es " | &strNombre | " y hoy es " | &dtHoy);
```

6.2.7.4 El operador especial @

El operador especial @ convierte un string que almacena el nombre de una definición de referencia (Definition Name Reference) en la propia definición. La principal utilidad es poder recuperar de BBDD referencias de definición almacenadas en modo string para utilizarlas en PeopleCode.

```
Local String &strNombreRegistro = "RECORD.CLIENTE";
Local record &recCliente;

recCliente = createRecord(@&strNombreRegistro);
```

Hay que tener cuidado a la hora de usar el operador @ debido a que si cambiamos el nombre de la definición de referencia, el string que almacene el nombre o el campo de BBDD con el nombre, no se actualizarán automáticamente pudiendo ocasionar problemas en la ejecución del programa.

6.2.7.5 Operadores de comparación

Estos operadores se utilizan para comparar dos expresiones que tengan el mismo tipo de datos devolviendo como resultado un tipo de dato booleano.

Operador	Significado
=	Igual a
!=	Distinto que
<>	Distinto que
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
Not	No (invierte el significado combinado con el resto de operadores). Ejemplos de uso : • Not= • Not< • Not >=

6.2.7.6 Operadores booleanos

En PeopleCode existen los siguientes operadores booleanos:

- AND: Operador Y / Intersección.

- OR: Operador O / Unión.
- NOT: Operador NO / Exclusión

Con dichos operadores se realiza la lógica booleana:

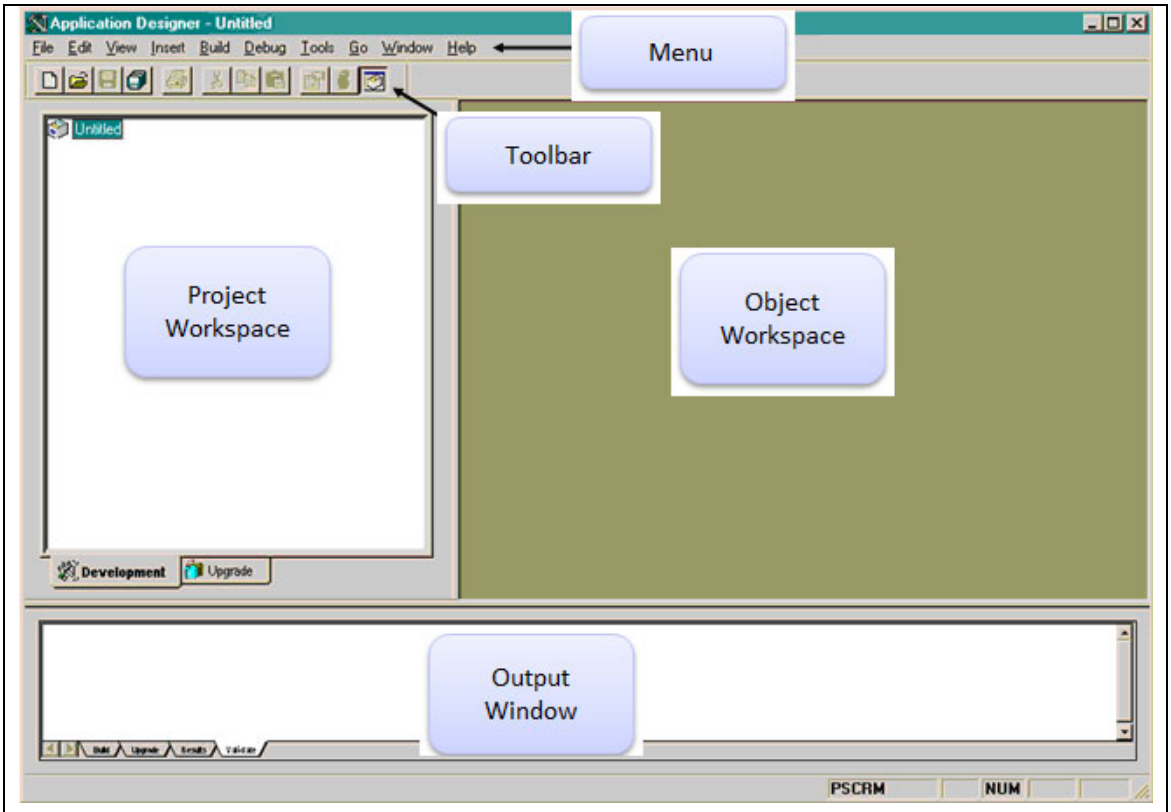
Expresión 1	Operador	Expresión 2	Resultado
FALSO	AND	FALSO	FALSO
FALSO	AND	VERDADERO	FALSO
VERDADERO	AND	VERDADERO	VERDADERO
FALSO	OR	FALSO	FALSO
FALSO	OR	VERDADERO	VERDADERO
VERDADERO	OR	VERDADERO	VERDADERO
FALSO	NOT		VERDADERO
VERDADERO	NOT		FALSO

Respecto a la precedencia en la evaluación, primero se evalúa el operador NOT, en segundo lugar el operador AND y por último el operador OR.

6.3 Introducción a Application Designer

Application Designer es la herramienta de programación que utiliza PeopleSoft para el desarrollo de los objetos y procesos que componen la aplicación. Se instala en modo cliente en los PCs habilitados para el desarrollo y se configura para conectarse a la base de datos servidor de PeopleSoft, en la que estarán almacenados todos los objetos y procesos predefinidos. En dicha base de datos se quedarán guardados todos los objetos nuevos desarrollados.

El aspecto gráfico de Application Designer es el siguiente:



Application Designer

Application Designer se compone de los siguientes elementos:

- Menu: Contiene las opciones principales de la herramienta (Archivo, Editar, Ver, Insertar, etc).
- Toolbar: Contiene botones con las funciones principales. La barra de herramientas cambia según el objeto que tengamos abierto.
- Object WorkSpace: Área de trabajo en la que crearemos o abriremos las definiciones de los objetos.
- Project: Área de desarrollo en la que se irán añadiendo todos los objetos de nuestro proyecto.
- Output Window (en la parte inferior de la pantalla): Área que muestra información relevante sobre validaciones, resultados, logs u otra información útil.

Los objetos que se pueden crear y manejar desde el Application Designer son los siguientes:

- Activity (actividad): Es un mapa que muestra un flujo de pasos, eventos y rutinas necesarios para completar una actividad en un proceso de negocio.
- Application Engine (motor de aplicación): Es el objeto fundamental de programación batch utilizado por PeopleSoft. Está compuesto de sentencias SQL y por el lenguaje de programación PeopleCode propio de PeopleSoft.
- Application Package (paquete de aplicación): Permite crear clases específicas para la utilización de programación orientada a objetos.
- Approval Rule Set (conjunto de reglas de aprobación): Mapas de flujos que permiten una representación visual de reglas de aprobación para transacciones.
- Business Interlink (interconexión de negocio): Permite una interconexión entre PeopleSoft y aplicaciones externas.
- Business Process (proceso de negocio): Mapas de flujos que permiten una representación visual de las actividades que componen un proceso.
- Component (componente): Es una transacción de lógica de negocio o un conjunto de páginas relacionadas procesadas conjuntamente.
- Component Interface (interfaz de componente): Permite el acceso externo al componente, encapsulando la lógica y métodos que se pueden invocar.
- Field (campo): Son datos individuales que introduce el usuario y se almacenan en base de datos, en una tabla o vista.
- File Layout (diseño de fichero): Es la definición o mapeo de un fichero para ser procesado. Identifica la localización de los campos con datos dentro del fichero.
- HTML : Objeto con código HTML que puede ser insertado en las páginas.

- Image (imagen): Objeto para almacenamiento y visualización de imágenes.
- Menu (menú): Permite el acceso a los componentes y las páginas contenidas en dichos componentes.
- Message (mensaje): Define los datos a insertar en los mensajes de aplicación en tiempo real.
- Message Channel (canal de mensaje): Se corresponde con la definición de grupos de mensajes con un tratamiento similar.
- Mobile Page (página móvil): Son páginas basadas en component interfaces sincronizables para visualización en dispositivos móviles (por ejemplo en tablets o smartphones).
- Page (página): Proveen la forma de introducir, visualizar y editar datos de forma online. El sistema es capaz de validar la entrada de datos, escribirlos en base de datos, recogerlos y visualizarlos.
- Project (proyecto): Colección definida por el usuario de definiciones relacionadas que han sido desarrolladas o adaptadas del estándar de PeopleSoft.
- Problem Type (tipo de problema): Usado en la optimización de registros y transacciones.
- Record (registro): Todos los datos residentes en PeopleSoft se almacenan en tablas o registros como parte de una base de datos relacional.
- SQL: Pueden ser programas enteros SQL o fragmentos de sentencias SQL para reutilización en diversos lugares del código.
- Style Sheet (hoja de estilo): Colección de estilos que pueden ser usados dentro de las páginas.

En este proyecto fin de carrera nos vamos a centrar en dos objetos como referencia que hacen uso de la mayoría del resto de objetos para poder explicar y comprender el

funcionamiento del Application Designer como herramienta de programación. Los objetos a utilizar serán la *Página* como componente fundamental de uso para el on-line y *Application Engine* como objeto fundamental de programación batch de PeopleSoft.

6.4 Creación páginas con Application Designer

6.4.1 Tablas de la Base de Datos de PeopleSoft

Como paso previo al diseño de cualquier aplicación en PeopleSoft es necesario explicar los distintos tipos de tablas que nos podemos encontrar en la base de datos de PeopleSoft. Se trata básicamente de tres tipos:

- **Tablas de catálogo del sistema**: Almacenan las características físicas de las tablas, vistas, columnas, índices, etc. Por ejemplo, las tablas del sistema de Oracle.
- **Tablas de PeopleTools**: Almacenan información relacionada con objetos, proceso del sistema, etc. Estas tablas guardan objetos como pueden ser por ejemplo los registros, páginas o menús. Suelen empezar por el prefijo “PS” sin subrayado. Ejemplo: PSRECDEFN.
- **Tablas de datos de la aplicación**: Contienen los datos con los que trabaja el usuario de la aplicación. Suelen empezar con el prefijo “PS_”, con subrayado. Ejemplo: PS_JOB.

6.4.2 Validación de datos

Los datos en PS se validan mediante las tablas de valores válidos y las tablas de *xlats*. Para introducir en un campo valores provenientes de tablas de valores válidos, se hace a mano o bien mediante el botón de la lupa  . Se pueden buscar valores en las tablas o bien introducirlos directamente.

Otra forma de validar valores es con la tabla de xlats (XLATTABLE). Se emplea como diccionario de datos para campos que no necesitan una tabla única para todos los valores posibles, típicamente porque son unos pocos valores. Los valores deben ser del tipo carácter, tener una longitud entre 1 y 4 caracteres y estar formado por un conjunto pequeño de valores

estáticos. Un campo típico sería el campo estándar EFF_STATUS.. El aspecto de un campo XLAT es:

Campo tipo XLAT

En la tabla xlats también se pueden guardar campos Si/No (el campo FIELDVALUE de XLATTABLE es Y ó N). Se suelen representar por una casilla de verificación.

Otro método de validación es la fecha efectiva (EFFDT). Permite mantener un historial de los datos introducidos en una tabla. Si una fila pasa a no ser válida, se inserta una nueva fila con una nueva fecha efectiva, y usar el campo de estado a fecha efectiva (EFF_STATUS) para poner la fila en estado *Inactivo*. El campo EFFDT suele ponerse por defecto como 1/1/1900

6.4.3 Pasos básicos para la creación de aplicaciones on-line

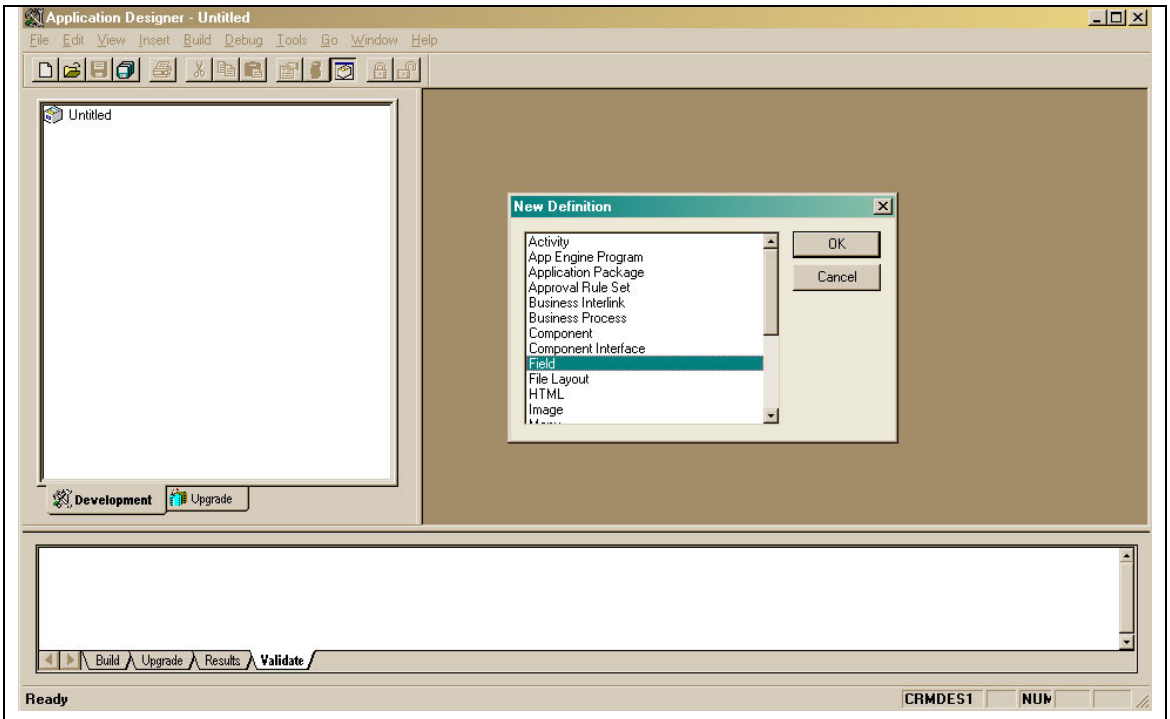
Para la creación de cualquier aplicación on-line en PeopleSoft podemos seguir una guía de 7 pasos fundamentales:

- Definición de campos nuevos
- Creación de definición de registro
- Creación de la tabla SQL.
- Creación de la definición de página
- Definición del componente
- Creación de definición de menú
- Activación de seguridad en menús

Una vez seguidos estos se pasos se procederá a publicar la aplicación/página desarrollada en la interfaz on-line de PeopleSoft lista para el uso de la misma. Para ello Application Designer provee un wizard de publicación que automatiza la misma.

6.4.3.1 Definición de campos nuevos en App Designer

Para definir los campos accederemos en el menú a la opción File > New > Field.



Creación de nuevo campo

Crearemos todos los campos necesarios como objetos aislados sin asignar a registros y dándoles las características principales como tipo, longitud y valores XLAT.

Los campos pueden ser:

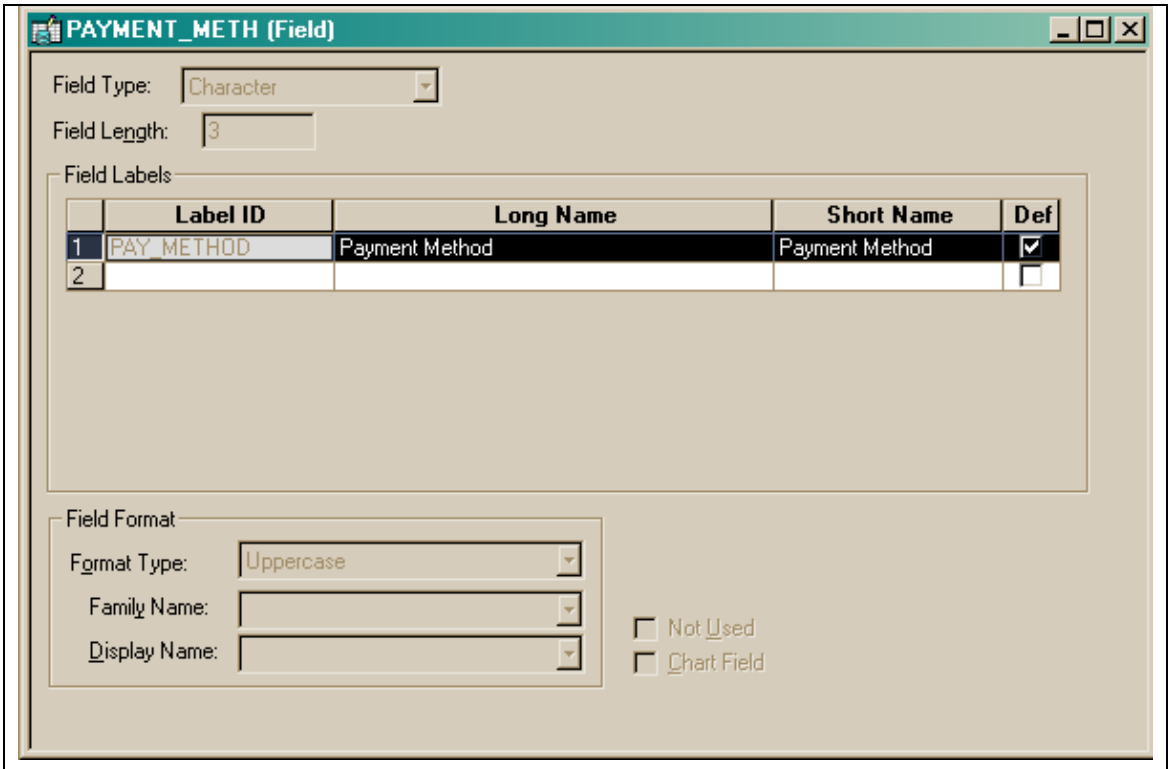
- De caracteres
- Numéricos
- Campos de fecha y hora: DATE, TIME y DATETIME.
- Campos de imagen.

Cuando se crea un nuevo campo o se modifica en el Application Designer, es conveniente comprobar la repercusión que el cambio puede tener. Para ello, se tiene el menú Edit > Find Object References, que lista los objetos que hacen referencia al campo abierto en este momento en el Application Designer. Esto es aplicable a más objetos, además de los campos. Por ejemplo, puede indicar que el campo está en dos records y además está usado en determinado PeopleCode. La pestaña Find Object References muestra la lista y haciendo doble clic sobre ella nos lleva al lugar donde se referencia el objeto.

Cuando se crea un campo XLAT, se pueden ver los valores del mismo en “File > Definition properties” o ALT+ENTER.

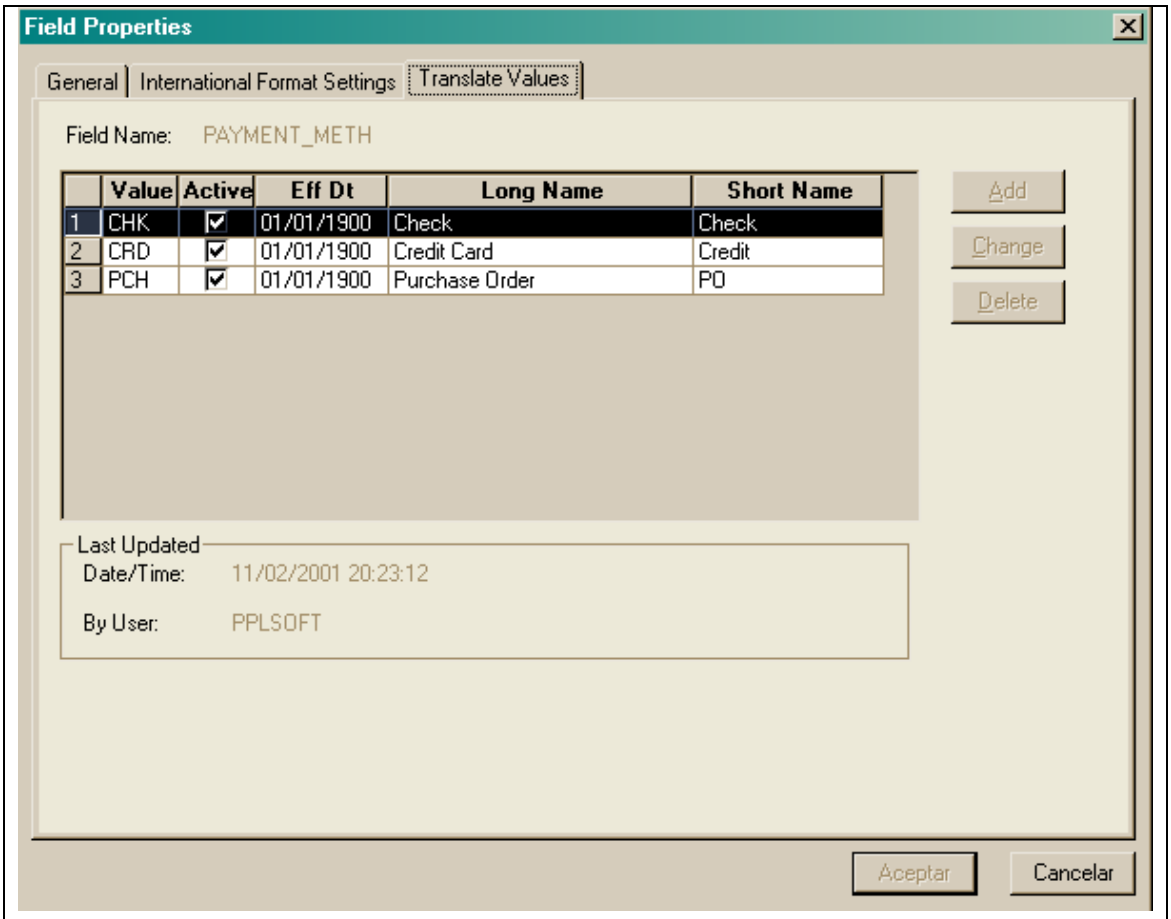
Cada vez que se crea un campo, o cualquier campo que se desee y esté en la BD, puede incorporarse al proyecto por medio de F7.

Ejemplo de campo:



Campo

Y sus propiedades:



Propiedades de campo

6.4.3.2 Creación de definición de registro

Cuando queremos definir una tabla SQL que agrupe los campos que hemos creado, se debe hacer mediante la creación de definiciones de registro en Application Designer. También se pueden crear vistas SQL, vistas de consulta, vistas dinámicas, subregistros y registros de trabajo.

Los campos creados se incluyen en la definición de registro. En este momento los campos tendrán dos tipos de atributos: atributos de campo, heredados de la definición de campo y atributos de campo en el registro, que definen como funciona el campo dentro del registro: por ejemplo: uso (claves, claves de búsqueda, etc.), validación (especificación de tablas de valores validos) y PeopleCode asociado.

Cuando se abre un registro tenemos los siguientes botones de vistas:

- **Field display:** Muestra los campos del registro y sus atributos.

- **Use Display:** Muestra características relacionadas con claves y valores por defecto del campo. Con el botón derecho sobre un campo, el menú Record Field Properties aparece con las características del campo en el record (si es clave de búsqueda, si tiene datos de una tabla de valores válidos, si es clave alterna de búsqueda, si tiene un valor por defecto, los controles de página, etc.)
- **Edits Display:** Muestra todas las reglas de validación de cada campo. El menú Record Field Properties es igual al de Use Display.
- **PeopleCode Display:** Contiene una columna para los eventos de PeopleCode e indica si dicho evento tiene código para ese campo.

Para añadir campos a un registro, hay que crear un registro nuevo y arrastrar y soltar los campos dentro de la ventana del registro.

Se pueden cambiar de orden los campos dentro del registro simplemente arrastrando y soltando. Además es posible mover campos entre dos definiciones de registro cortando y pegando.

También se pueden borrar los campos de los registros. En este caso, PeopleTools no elimina las referencias al mismo, es decir si por ejemplo hemos utilizado el campo en un página se mantendrá el mismo en la página aunque lo eliminemos del registro. Para evitar problemas, es necesario usar la opción Find Object References y así poder tener localizado el uso del campo. A demás hay que tener cuidado de eliminarlo en las páginas donde esté y de las tablas SQL con el comando ALTER correspondiente.

Cuando se guarda el registro por primera vez, se pide el Space Name: en Oracle coincidirá con el nombre del tablespace donde se creará la tabla correspondiente al registro.

6.4.3.3 Creación de la tabla SQL

Después de crear la definición de registro, y a partir de ella, se debe crear la tabla SQL (además de vistas, definiciones de índices, etc). Para ello hay que usar la función Build del Application Designer.

Se utiliza el proceso CREATE TABLE que hace lo siguiente:

1. Elimina la tabla si ya existe.

2. Crea la tabla de datos de la aplicación. El nombre es el de la definición del registro añadiendo “PS_”, con los nombres, tipos y longitud de columnas especificados en los campos del registro.
3. Marca los campos como NOT NULL.
4. Crea uno o varios índices: por los campos clave, por los campos de búsqueda alternativa, etc.

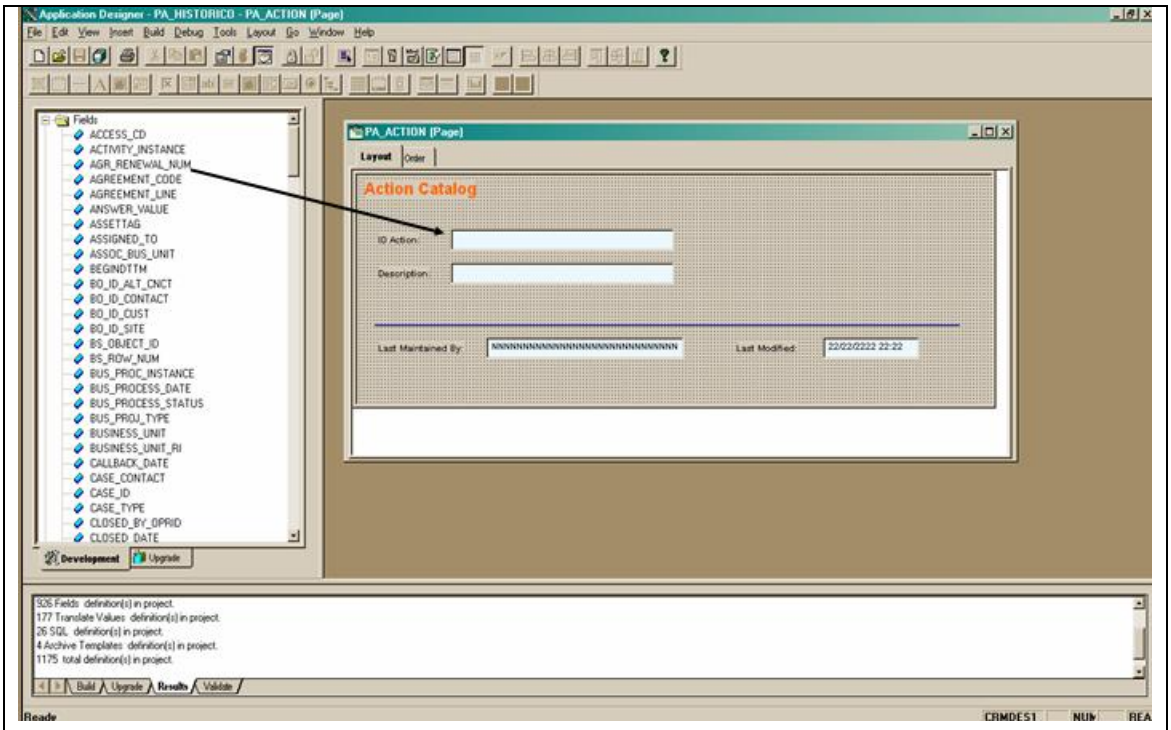
Para lanzar la creación de la tabla se utiliza el menú Build > Current Object... asegurándonos que el record está seleccionado.

6.4.3.4 Creación de definición de páginas

Una vez creada la tabla SQL, procedemos a crear la página incluyendo en ella los controles proporcionados por la barra de herramientas: botones, etiquetas, cuadros de texto, cuadros HTML, check box, drop-down list, etc.

La primera herramienta a tener en cuenta es el Object Inspector: menú View > Object Inspector. Gracias a esta herramienta podemos ubicar los controles de página en una posición exacta.

Los campos creados con anterioridad se introducirán en la página arrastrando y soltando desde la ventana de Project Workspace.



Inclusión de campo en página

Si abrimos un objeto página, con el botón derecho del ratón señalamos un campo, el menú Page Field Properties, vemos las propiedades del campo dentro de la página. Estas propiedades cambian según sea el tipo de campo: un botón de radio, un text field, un xlat, un campo con tabla de valores válidos, etc.

6.4.3.5 Definición de un componente

Cada página que aparece en un menú debe asociarse a un componente. Un componente determina una unidad transaccional, y las páginas del componente aparecen como fichas. También se determina en el componente las opciones y los campos de búsqueda que aparecen en el mismo.

Un componente se añade a un menú. Por tanto, desde el menú se accede al componente, que a su vez da acceso a las páginas.

Para crear un componente accederemos a: File > New... > Component. De la misma manera que los campos, se añaden páginas al componente arrastrando y soltando.

6.4.3.6 Definición de menú

Para añadir una nueva definición de menú necesitamos haber agrupado las páginas en componentes.

Para añadir un componente a una página del menú sólo hay que arrastrar el componente desde la ventana de la izquierda al menú.

Con esta opción le estaremos indicando a PeopleSoft dónde queremos publicar la página o componente dentro del árbol del portal de PeopleSoft.

Tras la creación de la definición de menú habría que ejecutar el “wizard” de publicación de páginas de Application Designer para que esté disponible en el portal.

6.4.3.7 Activación de la seguridad en menús

La seguridad al nivel de usuario se asigna mediante los conceptos de listas de permisos, roles y usuarios. Una lista de permisos define un grupo de páginas. A su vez, un rol puede tener varias listas de permisos, y un usuario, tener asignados varios roles:

- **Listas de permisos:** una lista de permisos puede incluir varios tipos de permisos: intervalos de desconexión, límite de tiempo, opciones de menú, etc.
- **Roles:** Vinculan las listas de permisos a los perfiles de usuarios.
- **Perfiles de usuario:** Conjuntos de datos que permiten describir a un determinado usuario de sistema PeopleSoft.

En PS también se puede definir seguridad al nivel de fila y al nivel de campo. La seguridad a nivel de fila se puede implementar mediante vistas SQL. La seguridad a nivel de campo se puede implantar mediante PeopleCode.

Esta configuración no se hace desde Application Designer sino que se hace desde el Portal de PeopleSoft en la sección de menú PeopleTools.

La seguridad en menús de PeopleSoft se define de la siguiente manera:

1. Se crea una lista de permisos: Inicio > PeopleTools > Mantenimiento de Seguridad > Uso > Listas Permisos y se elige el menú, los componentes y los permisos de acceso necesarios para las páginas.
2. Se crea un rol: Inicio > PeopleTools > Mantenimiento de Seguridad > Uso > Roles y se asigna al rol creado la lista de permisos creada en el paso 1.

3. Se incorpora el rol creado al perfil de usuario al que queremos dar permisos: Inicio > PeopleTools > Mantenimiento de Seguridad > Uso > Perfiles Usuario

6.5 Application Engine

6.5.1 Introducción a Application Engine

Application Engine es el componente de PeopleSoft concebido para lanzar aplicaciones batch y sobre el que se realizan los desarrollos de dichas aplicaciones. Está pensado, principalmente, para lanzar procesos intensivos en base de datos.

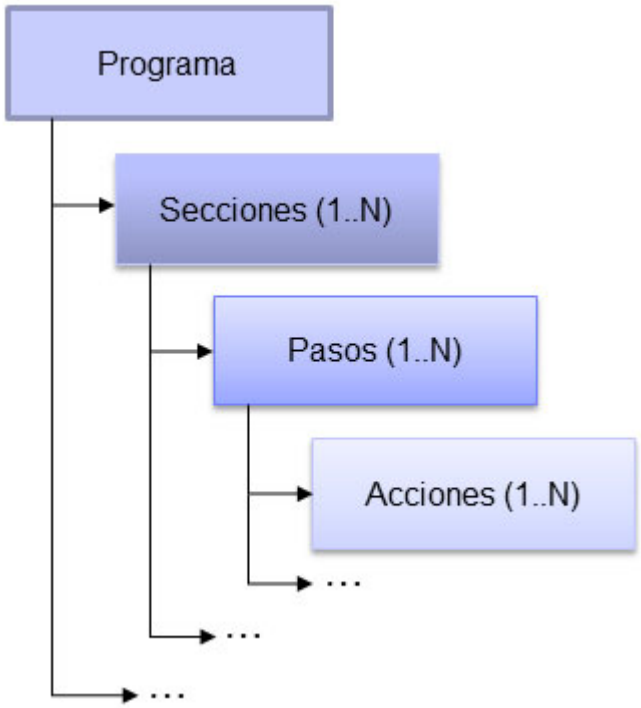
El programa (program) es la unidad ejecutable de Application Engine. Los programas se almacenan en la base de datos de PeopleSoft y se pueden crear y modificar mediante una interfaz gráfica integrada en la herramienta Application Designer de PeopleSoft. Los elementos que componen un programa de Application Engine se representan gráficamente en esta interfaz de acuerdo con la jerarquía existente entre unos elementos y otros.

Los Application Engines facilitan la reutilización de código permitiendo a un programa llamar a elementos existentes en otros programas, llamar a módulos escritos en PeopleCode, lo que permite acceder a la mayor parte de la arquitectura de integración de PeopleCode, e incluso llamar a módulos de otros lenguajes de programación.

Application Engine, junto con Process Scheduler (el planificador de procesos) proporciona los servicios necesarios en un entorno batch, si bien, dichos servicios están siempre enfocados al hecho de que se ha de trabajar en torno a la base de datos de PeopleSoft.

6.5.2 Elementos de un programa de Application Engine

Los elementos que componen un Application Engine pueden verse en el gráfico siguiente:



Elementos de Application Engine

6.5.2.1 Programa (Program)

El objeto programa es un módulo de programación, donde se puede incluir un programa en sí o bien secciones para ser utilizadas por otros programas.

Tiene que tener al menos una sección. La sección principal, aquella que ejecuta el programa desde el principio, debe tener obligatoriamente el nombre MAIN.

Un programa tiene asociadas varias propiedades y elementos. Entre ellos está el tipo de programa (batch only, library only y otros), los registros de estado asociados al programa, que veremos más adelante, y las tablas temporales que se asocian con cada programa. También se puede establecer si un programa tiene la característica de ser rearrancable o no.

De los elementos descritos a continuación, los elementos 'sección' , 'paso' y 'acción' son elementos que están definidos dentro de un programa. Los elementos 'registro de estado' y 'tabla temporal' son elementos definidos fuera del programa y declarados y utilizados dentro del programa.

6.5.2.2 Secciones (Sections)

Haciendo una analogía con otros lenguajes de programación, una sección viene a ser algo así como una subrutina, procedimiento o función.

Se compone de uno o más pasos. Todos los programas deben contener al menos una sección con nombre MAIN, salvo que el Application Engine sean de tipo “Library only” en cuyo caso no contendrá ninguna sección con el nombre MAIN.

En un programa de Application Engine, la sección MAIN es aquella en la cual se inicia la ejecución del programa.

Dentro de un programa puede haber una o varias versiones de una misma sección. Application Engine elige una versión u otra dependiendo del tipo de algunas propiedades que se asocian a esa versión. Por ejemplo, se puede establecer la fecha de activación de la sección, el mercado al que esté dirigida la sección y la plataforma (Oracle, SQL Server...) para la que esté hecha la sección. Application Engine elegirá una versión u otra a la hora de ejecutar la misma.

Las secciones pueden ser públicas, en cuyo caso podrán ser llamadas desde otros programas, o privadas, y en este caso solamente se podrán invocar desde el programa en el que están escritas.

6.5.2.3 Pasos (Steps)

Conforman la mínima unidad de trabajo desde la que se puede realizar un commit a base de datos. Esto quiere decir que todas las acciones de un paso están englobadas en la misma transacción de base de datos.

Los pasos de una sección se ejecutan secuencialmente.

Un paso se compone de una o varias acciones.

6.5.2.4 Acciones (Actions)

Típicamente cada una de las acciones en un paso realiza una tarea simple.

En tiempo de ejecución, el orden de ejecución de las acciones dentro de un paso depende del tipo de acción.

Cada una de las acciones puede ser de los siguientes tipos:

6.5.2.4.1 Do Actions

Manejan el flujo del paso en el que están incluidas dependiendo de las filas devueltas en una sentencia de tipo SELECT. Pueden ser del tipo:

- Do While

En esta acción se realizan las acciones que vienen a continuación mientras la sentencia Select contenida en la acción Do While devuelva filas. Es decir, se ejecuta la sentencia Select y si hay filas se ejecutan las acciones que vienen a continuación. Entonces se vuelve a ejecutar la sentencia, y si hay filas, de nuevo se ejecutan dichas acciones y así sucesivamente, hasta que no se devuelvan filas.

- Do When

Con esta acción se realizan las acciones siguientes si la sentencia Select contenida en la acción Do When devuelve filas. En este caso no funciona como un bucle. Sería parecido a un IF.

- Do Select

La acción de tipo Do Select hace que las acciones que vienen a continuación se ejecuten una vez por cada fila devuelta en la sentencia Select contenida en esta acción.

- Do Until

Similar a Do While, pero la sentencia Select se ejecuta después de haber ejecutado el resto de las acciones. Si hay filas sale del bucle.

6.5.2.4.2 SQL

Las acciones de tipo SQL permiten ejecutar sentencias SQL. Estas acciones contienen las siguientes sentencias:

- UPDATE
- DELETE
- INSERT
- SELECT

Hay que tener en cuenta que estas sentencias no sirven para controlar el flujo de ejecución a diferencia de las DoActions. Solamente ejecutan las acciones SQL que contienen.

6.5.2.4.3 PeopleCode

Permite ejecutar PeopleCode, el lenguaje de programación interpretado propio de PeopleSoft y hemos visto en el apartado 6.2 de este proyecto. Dentro del contexto de Application Engine podemos utilizar PeopleCode para lo siguiente::

- Componer sentencias SQL de manera dinámica. Este SQL se puede utilizar acciones de tipo SQL de Application Engine.
- Realizar control de flujo del programa. Se pueden insertar lógica de tipo IF/THEN/ELSE en PeopleCode y de acuerdo con el resultado, salir del paso o de la sección actual.
- Acceso a ficheros. PeopleCode es la manera estándar de acceder a ficheros en PeopleSoft. Cuando en un programa Application Engine se tenga que acceder a un fichero, se hará usando PeopleCode.
- Acceso a los distintos mecanismos de integración de PeopleSoft. Desde PeopleCode se puede acceder a mecanismos de integración tales como Business Interlinks e Integration Broker.
- Reutilización de Código. Desde una acción de tipo PeopleCode se puede acceder a funciones PeopleCode que se ejecuten también desde la interfaz de usuario. Además se puede acceder a Component Interfaces, que encapsulan la lógica de acceso a entidades de PeopleSoft reutilizando también la lógica usada en la interfaz on-line del usuario.

6.5.2.4.4 Log Message

Una acción de tipo Log Message permite escribir mensajes en la tabla estándar MESSAGE_LOG. Estos mensajes son multiidioma, están previamente parametrizados en la base de datos y pueden ser parametrizados y vistos desde una interfaz gráfica.

Este tipo de acción puede ser usado a modo de log de actividad.

6.5.2.4.5 Call Section

Una acción de tipo Call Section permite llamar a una sección localizada en el propio programa o a una sección pública de otro programa, incluida la sección MAIN de otro programa. La existencia de secciones y de acciones de tipo Call Section permite la reutilización de código de Application Engine y la estructuración del mismo en unidades lógicas.

6.5.2.5 Registros de Estado (State Records)

Los registros de estado son registros, definidos en Application Designer como cualquier otro registro de PeopleSoft, para almacenar temporalmente datos durante la ejecución de un programa.

Se pueden usar para pasar datos de unas secciones a otras y para pasar datos a un módulo PeopleCode.

Los hay de dos tipos, physical record y work record. Los de tipo físico se almacenan en la base de datos y deben contener de forma obligatoria el campo instancia del proceso (PROCESS_INSTANCE) como clave para que identifique correctamente la instancia de Application Engine que lo utiliza.

Un programa puede contener tantos registros de estado como sean necesarios.

6.5.2.6 Tablas temporales

Las tablas temporales permiten almacenar bloques de filas temporalmente en la base de datos. Las tablas temporales que va a utilizar un programa deben ser declaradas en éste, así como el número de instancias de un programa que pueden usar estas tablas en paralelo. PeopleSoft construye entonces tantas versiones o copias de la tabla temporal en la base de datos como sea necesario para que cada proceso acceda a una sola de estas versiones.

Las tablas temporales se han de usar como alternativa al almacenamiento de filas en la memoria del programa. Es un modo de que un conjunto de filas de base de datos pueda ser procesado sin que estas filas salgan de la base de datos.

6.6 Acceso a base de datos

6.6.1 Requerimientos de acceso a base de datos

Los procesos batch requieren de acceso a la base de datos de PeopleSoft CRM mediante sentencias SQL. El servicio de acceso a base de datos deberá estar optimizado y en la medida de lo posible ocultar las peculiaridades propias de la plataforma de base de datos utilizada.

Además se deberá proveer de la capacidad de dividir el trabajo en transacciones. En los casos en los que el proceso batch funcione de forma iterativa procesando un gran número de registros, se deberá ofrecer la posibilidad de parametrizar la frecuencia de commit de modo que se pueda afinar el comportamiento del proceso sin tener que realizar cambios en el código.

6.6.2 Implementación

Para el acceso a la base de datos de PeopleSoft CRM se utilizarán las posibilidades que ofrece Application Engine. Las maneras en las que se puede acceder a esta base de datos desde esta herramienta son dos:

1) Uso de las acciones de tipo SQL.

2) Uso de las acciones de tipo PeopleCode.

6.6.3 Acceso a base de datos desde acciones SQL

Este es el modo de acceso a base de datos estándar en Application Engine. Las sentencias incluidas en las acciones SQL incluyen además la posibilidad de insertar secciones en meta-SQL. Meta-SQL permite independizar al código SQL introducido de las peculiaridades de algunas plataformas.

Las acciones SQL de Application Engine tienen además opciones de optimización que permiten, por ejemplo, reutilizar cursores a base de datos o hacer inserciones en modo Bulk. La reutilización de cursores se puede realizar incluso después de haber realizado commit en la base de datos.

6.6.4 Acceso a base de datos desde PeopleCode

Desde las acciones de tipo PeopleCode también se puede realizar acceso a base de datos. Las utilidades de acceso en este lenguaje son bastante cómodas desde el punto de vista del programador. PeopleCode incluye algunos objetos tales como Record y RowSet que permiten acceder a la base de datos un modo sencillo y que formalmente se asemejan más a las entidades lógicas que representan. Estos objetos permiten acceder de un modo compacto, más mantenible y sin tener que escribir sentencias SQL.

Peso a ello, no se recomienda el uso del acceso de datos desde PeopleCode en procesos batch por no estar el acceso tan optimizado para este tipo de procesos como en el caso de las acciones SQL. Sin embargo, en algunos casos en los que el acceso a base de datos se ha de simultanear fila a fila con el acceso a ficheros o a otros elementos que son solamente accesibles desde PeopleCode, realizar el acceso a base de datos desde PeopleCode va a resultar más eficiente. En general, la importación o exportación de ficheros a la base de datos deberá hacerse íntegramente en PeopleCode.

6.6.5 Gestión de transacciones

La gestión de transacción se puede establecer a varios niveles. Hay que tener cuidado de que no haya ningún commit parametrizado en alguna sección o paso y que el desarrollador no espere, dando lugar a commits parciales que podrían producir inconsistencias si el resto del proceso falla. En general, un commit se ejecuta siempre que esté parametrizado en una acción o paso, nunca se ignora. La única excepción son las acciones de tipo Do Select, en las cuales se ignora el commit salvo que la acción sea un Do Select de tipo rearrancable.

En programas rearrancables, en el commit se realiza además el almacenamiento en base de datos de toda la información necesaria para que el proceso reanude la ejecución correctamente en caso de rearranque.

En cualquier caso, Application Engine no proporciona una funcionalidad de tipo rollback a la que llamar tras un error. La manera de volver para atrás una operación en base de datos de la que no se ha hecho commit es abortando la ejecución del programa batch. Esto puede dificultar en algún caso el tratamiento de errores.

6.6.5.1 A nivel de programa

Cuando un programa termina de forma correcta se realiza un commit a su finalización. Esta es la opción por defecto de realización de commit.

6.6.5.2 A nivel de sección

En la propiedad Auto Commit de una sección se puede establecer si una sección debe guardar la transacción tras ejecutar cada paso. Los pasos heredan esta propiedad (o la sobrescriben) y hacen commit en consecuencia.

6.6.5.3 A nivel de Paso

La propiedad Commit del paso admite los siguientes valores:

Default El paso hereda el nivel de commit de la sección en la que está incluido.

Later En este caso no se realiza el commit al finalizar el paso. Se espera a que se realice un commit en otro punto del programa. Se sobrescribe la opción elegida a nivel de sección.

Alter Step Se hace commit al terminar el paso, aunque en la sección tenga nivel de commit None.

Además si el paso contiene un bucle (una Do Action), la propiedad Frequency permite establecer la frecuencia de commit. Si esta propiedad es distinta de cero se hace commit cada N iteraciones y otro commit al terminar el bucle.

6.6.5.4 Desde PeopleCode

Desde PeopleCode se puede utilizar la sentencia CommitWork() para realizar un commit en base de datos. El commit se realiza independientemente de la parametrización de pasos y acciones. Si el programa es rearrancable, CommitWork() da lugar a un error en tiempo de ejecución ya que no permitiría la rearrancabilidad del mismo.

Si se invoca a una API desde PeopleCode, el método Save guarda los datos pero sin hacer commit. Se hace commit una vez que otro elemento del programa hace commit.

6.6.6 Buenas prácticas para el acceso a base de datos

En general de deberán utilizar acciones de tipo SQL para realizar los accesos a base de datos. Se utilizará PeopleCode en aquellos programas en los que se deba realizar tratamiento fila a fila de datos y para cada una de las filas se deba utilizar PeopleCode necesariamente, como es el caso de los accesos a ficheros o el acceso a Component Interfaces.

En lo que se refiere a la utilización del commit se ha de tener cuidado a la hora de colocar los puntos de commit. Siempre se ha de verificar que la situación de commit no conlleva peligro de inconsistencia en caso de un error posterior. También han de tenerse en cuenta aspectos relacionados con el rendimiento: commit demasiado frecuentes pueden hacer demasiado lento un proceso o por el contrario transacciones muy largas consumen muchos recursos de base de datos y pueden bloquear a otros procesos o incluso a los usuarios on-line.

La gestión del Commit desde PeopleCode solamente debe realizarse en casos de programas realizados íntegramente en PeopleCode y que no utilicen las opciones de rearrancabilidad proporcionadas por Application Engine.

Para aprovechar al máximo las posibilidades de las bases de datos relacionales, se recomienda utilizar el procesamiento basado en bloques. En general las sentencias SQL están optimizadas para trabajar con conjuntos o bloques de filas, y no con filas individuales. En el procesamiento basado en bloques se procura actualizar siempre bloques de filas en una única sentencia SQL. Se evita la situación en la cual se hace un bucle select/fetch y luego se actualiza fila a fila según las reglas de negocio a aplicar. Si se trabaja por bloques, se requiere de tablas temporales para almacenar datos de manera transitoria. Application Engine da soporte para el uso de tablas temporales.

6.7 Tratamiento de ficheros

6.7.1 Requerimientos en el tratamiento de ficheros

El uso de ficheros es especialmente importante para el tratamiento de datos masivos de tal forma que se puedan importar, exportar y hacer tratamiento de dichos datos. Además es uno de los métodos de integración batch que se usan con frecuencia, sobre todo en el caso de

interfaces con otros sistemas. Es por ello que desde los procesos batch se debe poder acceder en modo lectura o escritura a ficheros planos y Application Engine provee mecanismos para ello.

En el caso de procesos en los que se realicen gran número de transacciones y estén involucrados grandes ficheros, se debe tener un mecanismo que facilite la rearrancabilidad de estos procesos, pudiendo acceder en el rearranque a la posición de los ficheros coherente con el estado del proceso y de los datos actualizados en la base de datos.

6.7.2 Implementación

6.7.2.1 Acceso a ficheros

El medio utilizado para escribir y leer ficheros en Application Engine es siempre mediante PeopleCode. El objeto File de PeopleCode es el que da acceso a los ficheros.

Conjuntamente con el objeto File, el mecanismo FileLayout proporciona un medio automático y parametrizable de guardar o leer información en ficheros planos. Este mecanismo permite importar y exportar estructuras jerárquicas de datos en los formatos más comunes y con distintas posibilidades de formateo para fechas, cadenas de caracteres y números. Además, FileLayout utiliza objetos del tipo Record y Rowset que a su vez también se utilizan en el acceso a base de datos. Esto permite realizar de una manera sencilla y directa el intercambio de información entre la base de datos y ficheros.

6.7.2.2 Ficheros reposicionables

PeopleCode permite guardar o establecer la posición de escritura o de lectura de un fichero. Si esta posición se guarda en la base de datos, en caso de la interrupción anormal de un proceso, se podrá reposicionar el fichero en el punto de la última ejecución.

Para utilizar esta opción se debe abrir el fichero en modo Update. El fichero debe utilizar el juego de caracteres ANSI.

6.7.2.3 Buenas prácticas en el tratamiento de ficheros

6.7.2.3.1 Importación y exportación de datos

Por motivos de rendimiento, en los programas en los que se incorpore información leída de un fichero a la base de datos o viceversa, que típicamente incluye lectura de un registro del fichero o escritura (inserción o actualización) del registro en la base de datos, es aconsejable realizarlo íntegramente en la misma rutina de PeopleCode. Si se utiliza el mecanismo de rearranques de Application Engine y el bucle de lectura/actualización es muy largo, hay que realizar algunas consideraciones adicionales que se verán en el apartado dedicado a la arquitectura de rearranques.

En el caso de que se trate de lectura en un fichero e inserción en BBDD se usará el objeto SQL de PeopleCode. A la propiedad BulkWork de este objeto se le asignará el valor True. Esto hace que las inserciones no se manden una por una a la base de datos sino que manden en bloques. La mejora en rendimiento es notable en este caso. Habrá que tener cuidado y controlar la posible inserción de registros duplicados, ya que en este caso la inserción no devuelve error.

6.8 Gestión de errores

6.8.1 Requerimientos en el tratamiento de errores

Cualquier arquitectura debe proveer mecanismos para detectar errores y poder actuar de modo que no se generen inconsistencias en la base de datos o en la información enviada a otros sistemas. Los programas de Application Engine deben dar cobertura a este requisito básico de gestión.

6.8.2 Implementación

Application Engine permite de forma estándar la detección de errores para su tratamiento en algunos casos, y en otros aborta automáticamente el proceso. El desarrollador puede establecer condiciones que hagan que el programa se comporte de una manera dependiendo de la existencia de error mediante las propiedades habilitadas para tal efecto en los pasos y acciones de Application Engine

En cualquier caso, con este mecanismo estándar, no existe una función de tipo rollback que permita deshacer los últimos cambios en base de datos sin interrumpir la ejecución de proceso batch.

6.8.2.1 Tratamiento de errores a nivel de paso

La propiedad On Error permite decidir qué se ha de hacer en caso de error en SQL o en PeopleCode. Si se trata de un error de compilación de SQL, siempre se aborta la ejecución. Las opciones de las que se dispone son:

Abort El proceso termina con un mensaje de error.

Ignore El proceso continua, pero deja un mensaje de advertencia

Suppress El proceso continua y no se deja mensaje alguno

6.8.2.2 Tratamiento de errores a nivel de acción

La propiedad On Return de una acción de tipo PeopleCode permite, si el fragmento de código PeopleCode devuelve algo distinto de cero, realizar una de las siguientes acciones.

Abort El programa termina inmediatamente.

Break El programa sale de la sección en ejecución y el control vuelve a paso llamador.

Skip Step El programa sale del paso en ejecución y la ejecución se reanuda en el siguiente paso. Si se trata del último paso de una sección, el paso llamador retoma el control de la ejecución.

Esta utilidad se puede usar también para controlar el flujo de la aplicación desde una acción usando PeopleCode.

La propiedad No Rows de una acción de tipo SQL permite en caso de que una sentencia SQL afecte a 0 filas lo siguiente:

Abort El programa finaliza.

Section Break Application Engine sale de la sección en ejecución inmediatamente y el control pasa al paso que hizo la llamada.

Continue Continúa la ejecución del programa.

Skip Step	Application Engine sale del paso en ejecución y prosigue con el siguiente paso. Application Engine ignora el commit del paso en ejecución. Si el paso contiene solamente esta acción, se debe usar Skip Step para evitar el commit a nivel de paso.

6.8.2.3 Tratamiento de errores a nivel de PeopleCode

En general, los métodos y funciones en PeopleCode devuelven valores específicos en caso de error. Hay que prestar especial atención a los métodos y funciones que incluyen operaciones de entrada-salida.

- Los métodos de acceso a base de datos devuelven False si se ha producido un error.
- En caso de error en el acceso a un fichero, la rutina de PeopleCode termina con error.
- Los objetos de tipo FileLayout contienen una propiedad que permite saber si la última operación ha terminado con error.
- Los objetos de tipo Component Interface disponen además de objetos específicos para recuperar el mensaje de error.

Además, con PeopleCode se puede utilizar la cláusula try-catch para gestionar errores y excepciones.

6.8.2.4 Buenas prácticas en la gestión de errores

Cuando diseñamos aplicaciones, siempre se debe tener en cuenta que pueda ocurrir algún un error en cualquier punto del programa y evaluar cómo se ha de proceder en cada uno de los casos. Se ha de poner especial atención a las operaciones de entrada/salida, (acceso a base de datos, a ficheros, a mecanismos de integración...), ya que son las que más propensas a fallar por causas ajenas al propio programa.

Según la naturaleza del proceso se ha de evaluar si se debe utilizar la arquitectura de rearranques de Application Engine o no, para que en caso de error o finalización anormal de programas se pueda recuperar el proceso de una manera consistente.

Según la criticidad del proceso que se lleve a cabo, se deben diseñar los procesos para poder revertir a la situación inicial en el caso de que los datos suministrados sean erróneos. Ello hará que los procesos sean más tolerantes a los fallos.

En caso de error, conviene suministrar toda la información posible, de modo que la causa del error pueda ser establecida lo más rápidamente posible. Suministrar información durante toda la ejecución del proceso, en forma de trazas, ayuda además a investigar acerca de los errores cuya causa última pueda no estar clara.

6.9 Rearranques

6.9.1 Rearrancabilidad

La rearrancabilidad es una de las propiedades que tienen los programas Application Engine que permite a los mismos continuar ejecutándose desde el mismo punto en el que falló. Esta característica añade un plus importante en la usabilidad de Application Engine y por tanto la vamos con más detalle en este apartado.

6.9.1.1 Terminología

A continuación se indica una breve descripción de la terminología más frecuente en procesos rearrancables:

Término	***Descripción***
Programa rearrancable	Un programa es rearrancable si es capaz de continuar el proceso en el estado en que se encontraba antes de una terminación anormal. Debe estar organizado en Unidades Lógicas de Trabajo
Unidad Lógica de Trabajo (ULT)	Conjunto de actualizaciones relacionadas en la Base de Datos que dejan ésta en un estado coherente
Frecuencia de commit	Número de Unidades Lógicas de Trabajo que se procesarán antes de hacer un commit a la Base de Datos.
Puntos de commit	Serie de estados estables alcanzados por el programa (el proceso siempre se rearranca desde el último punto de commit)

Estado del programa	Información que permite el rearranque de su proceso desde el mismo punto en que se almacenó dicho estado (que será el último punto de commit)
Error Leve	Un error de procesamiento detectado en el programa pero que no impide continuar el proceso (también llamado Error Tratable). Ejemplo: Un valor incorrecto en un campo del registro leído → Acción: ignorar el registro y continuar con el siguiente.
Error Grave	Un error que no permite continuar el proceso normal del programa. Puede estar causado por una acumulación de errores leves. Ejemplo: No se puede obtener la fecha del sistema → Acción: interrumpir la ejecución e informar al operador
Estado del programa	En función del resultado de la ejecución anterior, un programa rearrancable podrá iniciar su siguiente ejecución en uno de los siguientes estados: Normal: Arranque normal. Se iniciará el proceso desde el principio Rearranque: La ejecución anterior terminó anormalmente. El programa debe continuar la ejecución desde el último punto de commit
Reposicionamiento de ficheros	Un programa rearrancado debe reposicionar sus ficheros de entrada y salida para asegurar que éstos se mantienen sincronizados con los cambios en la Base de Datos (es decir, restaurar el estado del último Punto de Commit)
Instancia de Proceso	Número que identifica una ejecución en concreto de un programa. Cuando un programa rearranca después de una parada anormal, sigue teniendo la misma instancia de proceso.

6.9.1.2 Concepto

La funcionalidad de rearranques permite que un proceso batch que terminó anormalmente por cualquier causa, se pueda iniciar de nuevo reanudando el procesado de datos en el punto inmediatamente posterior al último punto de commit realizado.

Algunos procesos son por naturaleza rearrancables y no requieren de ningún tipo de soporte especial. Por ejemplo, un proceso pequeño que permita realizar todas las operaciones en una única transacción sería conceptualmente rearrancable ya que si el proceso termina anormalmente la transacción no llega a efectuarse y el programa podría simplemente empezar desde el principio. Otro ejemplo de proceso rearrancable sería aquel que listara un conjunto de filas por procesar y cada vez que procesara una la dejara marcada como procesada en la base de datos. Si el programa termina anormalmente, al arrancar de nuevo no seleccionará las filas ya procesadas y por lo tanto el proceso realizará bien su trabajo sin necesidad de tener nada más en cuenta.

Sin embargo hay procesos que si se arrancan desde el principio, sin más, pueden afectar a la integridad de los datos, al duplicar acciones (inserts duplicados, por ejemplo). En tales casos se debe proveer un mecanismo que permita que el programa sepa que está arrancando de nuevo y pueda conservar su estado, la información suficiente para situarse en el último punto de commit de una manera consistente.

El desarrollador tendrá que dividir su programa en unidades lógicas de trabajo y hacer que las variables que contengan el estado del programa se guarden en base de datos cada vez que se realice un commit. En el caso en el que al mismo tiempo se esté procesando un fichero, habrá que saber cuál es el registro del fichero que se corresponden con el último punto de commit, para así comenzar la lectura o la escritura en ese punto, para ello los ficheros deberán ser reposicionables.

6.9.2 Requerimientos de rearrancabilidad

La arquitectura batch deberá dar soporte a la rearrancabilidad de procesos. Se deberá al menos dar un mecanismo que permita recuperar fácilmente el estado del programa. La rearrancabilidad afectará también a ficheros.

Además se podrá variar la frecuencia de commit en procesos iterativos.

6.9.3 Implementación

Application Engine proporciona soporte a procesos batch rearrancables. La principal característica que hace esto posible es el uso de registros de estado físicos, es decir, registros

que se guardan en base de datos y la capacidad de Application Engine de reanudar la ejecución en el paso siguiente al último punto de commit realizado.

Cada vez que Application Engine realiza un commit con el rearranque habilitado, guarda en la base de datos el estado del programa. Es decir, ejecuta un punto de commit.

La capacidad de Application Engine para recordar qué pasos se han completado y cuáles no dependen de la información guardada en el registro estándar AERUNCONTROL, cuya clave primaria es la instancia de proceso.

En un programa en ejecución con el rearranque habilitado, cada vez que se ejecuta un punto de commit también se guarda la información necesaria para que el programa rearranque en el registro AERUNCONTROL.

En cuanto a la información guardada en registros de estado, el programa es capaz de recordar aquellos registros de programa que son de tipo físico (que se guardan en una tabla de la base de datos).

Al reanudar la ejecución mediante línea de comando, se ha de especificar la instancia de proceso a ejecutar. El programa entonces retomará el valor de los registros físicos de las tablas correspondientes y reanudará la ejecución en el paso siguiente al del último punto de commit. Las variables globales de PeopleCode también son incluidas en el punto de commit y su valor es restaurado tras el rearranque.

Cuando el programa es rearrancable, no se permite hacer commit desde PeopleCode. De este modo, una llamada a una acción de este tipo queda siempre englobada toda entera dentro de una transacción.

6.9.3.1 Características de Application Engine en programas rearrancables

6.9.3.1.1 A nivel de programa

Application Engine guarda el registro de estado cada vez que hay un commit a lo largo del programa si éste ha sido habilitado para admitir rearranque. Un programa se habilita para que sea rearrancable mediante la línea de comando. También mediante las propiedades del programa, en Application Designer, se puede establecer si el programa pude ser rearrancable o no.

6.9.3.1.2 A nivel de sección (*section*).

Una de las propiedades que se pueden asociar a una sección es el tipo de sección (section type). Se puede decir si una sección es crítica (actualiza datos de carácter permanente en la base de datos) o es solamente de preparación. El campo AE_CRITICAL_PHASE del registro AERUNCONTROL (el que conserva el estado del programa) toma el valor Y desde el momento en que se ejecuta la primera sección crítica. Esto permite saber si el programa, en caso de rearranque, puede empezar normalmente desde el principio o se debe reposicionar en el punto adecuado porque ya se ha alterado información crítica antes del último commit.

6.9.3.1.3 A nivel de paso (*step*).

La cláusula Do Select , que permite realizar un paso de un bucle para cada fila obtenida de un select puede ser de tipo rearrancable (restartable type), lo que permite que se puedan realizar commits intermedios antes de haber terminado de tratar todas las filas. Esto permite tratar un gran número de filas en varias transacciones en lugar de hacerlo en una sola.

El desarrollador tendrá que componer las sentencias SQL de modo que sean rearrancables. Por ejemplo, la sentencia:

```
SELECT RECNAME, FIELDNAME
   FROM PS_AERECFIELD
   ORDER BY RECNAME, FIELDNAME
```

debería ser reescrita de la siguiente forma:

```
SELECT RECNAME, FIELDNAME
   FROM PS_AE_RECFIELD
   WHERE RECNAME > %Bind(RECNAME)
   OR (RECNAME = %Bind(RECNAME) AND FIELDNAME >
   %Bind(FIELDNAME))
   ORDER BY RECNAME, FIELDNAME
```

y en este ejemplo, los campos RECNAME y FIELDNAME deberían guardarse en el registro de estado, de modo que puedan recuperarse el valor correspondiente con el último commit tras un error del programa (utilizando la palabra clave %Bind). De este modo se recuperan solamente las filas que todavía no han sido procesadas.

6.9.3.2 Ficheros reposicionables

La forma de utilizar ficheros reposicionables es diseñar los programas de manera que guardemos en un registro de estado de la base de datos la posición del fichero en la que se encuentra el fichero justo a la hora de hacer commit. En caso de rearranque, se deberá abrir el fichero e ir a la posición almacenada en base de datos. PeopleCode ofrece utilidades para conseguir la posición en la que se encuentra un fichero y para situarse en esa posición.

La clase File posee los métodos GetPosition y SetPosition que como su nombre indica permiten obtener la posición actual en la que estamos leyendo o escribiendo en un fichero y situarnos en una posición en concreto, respectivamente.

La posición es un número entero cuyo valor es reservado, es decir, el valor obtenido con GetPosition solamente puede ser utilizado con la función SetPosition. El único valor que se puede introducir en la función SetPosition que no haya sido obtenido mediante GetPosition es el valor 0, para situarnos el principio de un fichero.

Para poder usar GetPosition y SetPosition, hay que abrir el fichero en modo actualización (flag “U”)

Ejemplo:

```
&MYFILE.Open(&SOMENAME, "U");
If &MYFILE.IsOpen Then
   while &MYFILE.ReadLine(&SOMESTRING)
      &CURPOS = &MYFILE.GetPosition();
      /* Guardar el valor de &CURPOS después de cada lectura,
         y procesar el contenido de  &SOMESTRING */
   End-While;
End-If;
&MYFILE.Close();
```

6.9.3.3 Programas rearrancables con reposicionamiento de ficheros

Esta sección presenta varias posibilidades que podrían tenerse en cuenta para implementar procesos rearrancables cuando hay que realizar operaciones con ficheros utilizando Application Engine.

La parte principal de este tipo de programas suele ser un bucle. En cada iteración de este bucle se suele procesar un registro de un fichero y se insertan, actualizan o se seleccionan uno o varios registros de base de datos.

Cuando se trata de un número elevado de filas a tratar conviene poder realizar un punto de commit antes de tratar todas las filas. Lo normal es hacer uno de estos commit cada n filas. Para poder realizar este commit periódico y poder rearrancar el proceso en caso de interrupción, se plantean las siguientes posibilidades:

1) Una posibilidad podría ser iterar pasos del Application Engine, utilizando una sentencia de tipo Do Select rearrancable, y en cada uno de estos pasos introducir acciones de tipo PeopleCode en las que se realizará el volcado de una fila a fichero o la lectura de una fila desde fichero (PeopleCode es la única manera de acceder a ficheros desde Application Engine). Estos fragmentos de PeopleCode grabarían en los registros de estado la posición en la que han quedado los ficheros tras la lectura o la escritura en ellos. Esta opción permitiría controlar mediante las utilidades que proporciona Application Engine tanto los rearranques con la frecuencia de commit.

   Sin embargo, realizar llamadas a PeopleCode dentro de un bucle de Application Engine cuando el bucle tiene muchos pasos es muy costoso, ya que en cada iteración se ha de arrancar y parar el intérprete de PeopleCode.

2) Una posibilidad que permitiría seguir utilizando las posibilidades de rearranque que da Application Engine sería que las acciones en PeopleCode procesaran cada iteración un rango de registros amplio.

   Como resultado no habría tantas llamadas a código PeopleCode. Dentro de PeopleCode habría código de acceso a base de datos y de acceso a ficheros, pero los commit se ejecutarían externamente al bloque de PeopleCode.

3) El proceso se podría realizar en una rutina de PeopleCode incluyendo los commit. Esta opción puede ser problemática debido a que el commit realizado desde PeopleCode es incompatible con las opciones de rearrancabilidad de Application Engine. Por ello

habría que montar una infraestructura parecida a la que ya dispone Application Engine para los rearranques, pero que se debería manejar desde PeopleCode.

La primera posibilidad es la más viable y la que se establece como mecanismo estándar para PeopleSoft. La segunda y tercera posibilidades se discuten en los siguientes puntos:

6.9.3.3.1 Estructura de un programa que procese en PeopleCode un bloque de registros cada vez

Se pretende con este esquema procesar en PeopleCode un bloque de registros, de modo que no se arranque el intérprete de PeopleCode cada vez que se procese un registro. De este modo se podría conseguir hacer uso de la rearrancabilidad que ofrece Application Engine sin penalizar mucho el rendimiento. El programa podría tener el siguiente aspecto:

Programa que lea de un fichero:

1) Paso con acción SQL que consiga los datos de configuración. Los datos se introducirán en un registro de estado de tipo físico. Uno de los campos del registro de estado contendría la frecuencia de commit, que se aplicará en el bucle contenido dentro de la rutina de PeopleCode.

2) Paso con un bucle, de tipo while . Dentro del bucle habrá una acción de tipo PeopleCode que realice la importación del fichero.

   Este paso se puede parametrizar para que tenga una frecuencia de commit 1, es decir, que cada vez se ejecute el bloque de PeopleCode se realice un punto de commit. Hay que tener en cuenta que a su vez la rutina de PeopleCode va a procesar n registros.

   La rutina PeopleCode tendrá la siguiente secuencia de acciones:

   2.1) Apertura y reposicionamiento de ficheros: Habría que considerar la posibilidad de mantener un fichero abierto a través de sucesivas llamadas al programa en PeopleCode utilizando una variable global. El

reposicionamiento se realizará llamando al método SetPosition. La posición de los ficheros se ha de obtener de los registros de estado.

2.2) Realizar n pasos del bucle, realizando la importación de datos. Es decir, en cada interacción del bucle se leería de fichero y se haría insert o update en base de datos.

2.3) Guardar la posición de los ficheros en los registros de estado. La posición se obtiene usando el método GetPosition.

2.4) Si el fichero se ha acabado, retornar 1. De acuerdo con esta condición, el programa debería salir de bucle principal.

3) Finalización del programa Application Engine

Cada vez que el programa realizara un commit, se guardaría la posición de los ficheros, lo cual permitiría situarse en el punto adecuado en caso de rearranque.

Programa que escriba en un fichero:

Se podrían establecer los siguientes pasos:

1) Paso con acción SQL que consiga los datos de configuración. Los datos se introducirán en un registro de estado de tipo físico. Uno de los campos del registro de estado contendría la frecuencia de commit, que se aplicará en el bucle.

2) Paso con un bucle, de tipo while . Dentro del bucle habrá una acción de tipo PeopleCode que realice la exportación del fichero. El commit se puede realizar a nivel de este paso. El programa PeopleCode tendrá la siguiente estructura:

2.1) Apertura y reposicionamiento de ficheros: Habría que considerar la posibilidad de mantener un fichero abierto a través de sucesivas llamadas al programa en PeopleCode. El reposicionamiento se realizará llamando al método SetPosition.

2.2) Realizar n pasos del bucle, realizando la exportación de datos. El bucle seguramente irá guiado por una sentencia SQL de tipo select, ejecutada dentro de PeopleCode. Dicha sentencia deberá ser tal que seleccione un bloque de registros y que sea rearrancable.

2.3) Guardar la posición de los ficheros, usando el método GetPosition.

2.4) Guardar los campos necesarios para que la siguiente vez en que se invoque este programa, se seleccione el siguiente bloque

2.5) Si ya se han exportado todos los registros, retornar 1. De acuerdo con esta condición, el programa debería salir de bucle principal.

3) Finalización del programa Application Engine.

6.9.3.3.2 Rerrancabilidad manejada desde PeopleCode

En este apartado vamos a ver cómo podría ser un programa de Application Engine en el que el commit y la rearrancabilidad se manejaran desde PeopleCode. El programa sería de tipo no rearrancable desde el punto de vista del Application Engine, para que el commit se pueda hacer desde PeopleCode.

La estructura global del programa podría ser como sigue.

1) Un paso en el que desde una acción de tipo SELECT se consiguieran parámetros tales como Run Control Id, el nombre del programa y otros parámetros de configuración.

2) Un paso con una acción de tipo PeopleCode que realice todo el cuerpo del programa. Esta acción tendría los siguientes pasos

2.1) Inicio del programa:

Averiguar si el programa está empezando desde el principio. Para ello se consultaría una tabla de rearranques, que sería general para todos los programas que usaran esta utilidad. Esta tabla tendría como clave los campos

nombre de programa y Run Control ID. Tendría un campo con el estado, que sería 'en ejecución' o 'listo'.

Si el programa está en modo rearranque ('en ejecución'), habría que conseguir los datos que nos permitan reanudar adecuadamente el programa. Estos datos deberán estar en un registro específico de cada programa. El registro estará mapeado a una tabla física y tendrá como clave los campos Run Control ID y Program Name. Contendría los campos necesarios para ser capaces de situarnos en el punto de ejecución correcto del programa, por ejemplo, la posición que tenían los ficheros en el último commit.

En cualquier caso habrá que actualizar el estado del programa para que quede 'en ejecución'.

Si el programa está en modo arranque normal, habrá que realizar una primera inicialización del registro de estado (el registro con los datos de rearranque) y hacer una inserción de éste en la base de datos.

2.2) Apertura de Ficheros.

Abrir los ficheros que se hayan de utilizar en el proceso de datos

Si el programa está en rearranque, ir a la posición que indiquen los registros de estado.

2.3) Cuerpo del programa

El programa debería estar claramente divido en unidades lógicas de proceso de modo que sepamos las que hay que procesar y las que no en caso de rearranque.

Bucle. Es muy posible que la mayor parte del procesado se realice en un bucle. Cada n iteraciones del bucle se tendrá que realizar un commit. La condición sobre la que se haga el bucle se habrá de diseñar de modo que sea rearrancable.

Commit. El commit¸ además del commit en base de datos, deberá incluir de incluir los pasos necesarios para guardar todos los datos relevantes al rearranque. En particular será importante guardar la posición de los ficheros, para poder volver a esa posición en cualquier momento.

2.4) Fin del programa

En el fin de programa se debe actualizar en el registro de rearranque para dejarlo en estado 'listo' el estado de programa y borrar las filas con el registro de estado.

6.10 Seguridad y accesibilidad

6.10.1 Implementación

Hay que tener en cuenta distintos aspectos de seguridad tanto en tiempo de desarrollo como en tiempo de ejecución.

6.10.1.1 En tiempo de desarrollo

En general, los objetos accesibles desde Application Designer se pueden agrupar en definiciones (Definitions) y se asignan permisos a los desarrolladores para poder modificarlos o no.

Adicionalmente, las secciones de los programas del Application Engine se pueden proteger para que no puedan ser usadas por otros programas.

6.10.1.2 En tiempo de ejecución

La ejecución de programas del Application Engine está protegida mediante el uso de usuario y contraseña.

En la herramienta Process Scheduler se puede dar permisos a los usuarios on-line para poder ejecutar o denegar explícitamente la ejecución de procesos de Application Engine.

En cualquier caso, a lo hora de ejecutar estos procesos mediante línea de comando, aplica también la seguridad y los permisos Unix, que permite a ciertos usuario Unix ejecutar procesos.

6.11 Ejecución de programas App Engine

Es posible ejecutar programas Application Engine de tres formas distintas:

- Modo batch: A través del Process Scheduler.
- Modo manual: A través de la línea de comandos con el comando psae.exe
- Modo online: A través de PeopleCode con la sentencia CallAppEngine.

6.11.1 Modo batch: Process Scheduler

6.11.1.1 Run Control ID

Para ejecutar un proceso se debe informar al sistema sobre cuándo y dónde queremos que el proceso se ejecute.

Un run control es un registro de la base de datos que proporciona valores para estas configuraciones. En vez de entrar los mismos valores cada vez que ejecutamos un proceso, podemos crear y grabar un run control con estas configuraciones. La siguiente vez que ejecutemos el proceso, seleccionamos el run control, y el sistema rellena las configuraciones.

Podemos crear run controls que apliquen a varios procesos relacionados.

Un run control ID es usado como una clave (con un ID de usuario) para los registros que contienen los parámetros que un proceso necesita en tiempo de ejecución. Al almacenar los parámetros en una tabla que el proceso puede consultar usando el run control ID y el ID de usuario se capacita al proceso para poder ser ejecutado sin la intervención del usuario.

Los run control IDs son almacenados como mínimo en dos tablas: una tabla de control de ejecución de aplicaciones (PS_PRCSRUNCNTL) y una tabla de control de ejecución de herramientas (PSPRCSRQST), que almacena información requerida por el Process Scheduler.

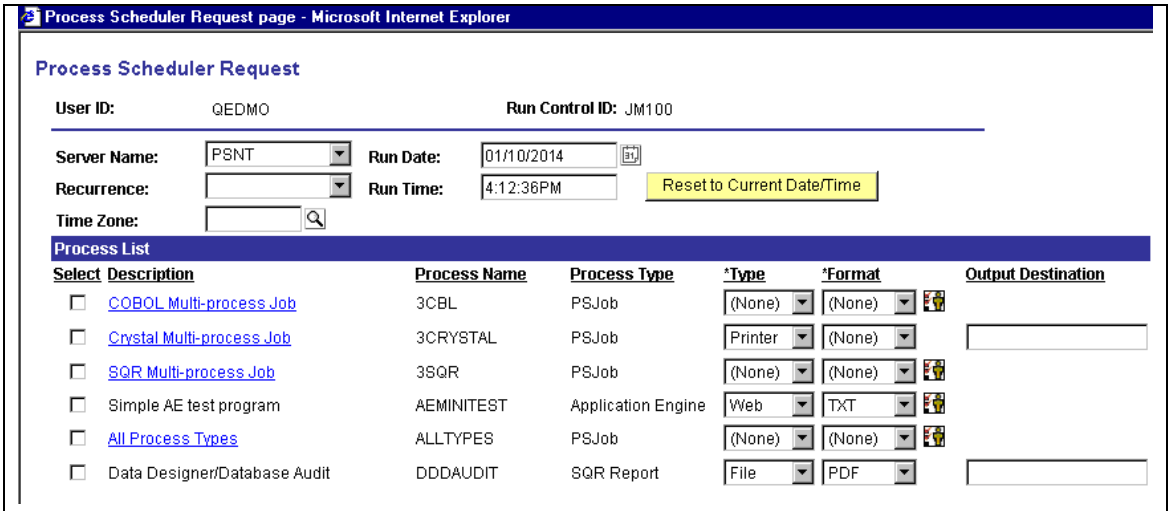
6.11.1.2 Solicitud de proceso

Para realizar una solicitud de proceso se debe acceder en el portal de PeopleSoft al siguiente menú: PeopleTools > Process Scheduler > System Process Request.

Esta página nos permite:

- Añadir un nuevo run control ID
- Seleccionar un run control ID existente

La página de Process Request Dialog muestra el run control ID que hemos seleccionado o añadido para realizar la solicitud de proceso. Se pulsa el botón Run para acceder a la página Process Scheduler Request, que muestra todos los jobs y procesos (definidos previamente) que tenemos seguridad para poder ejecutar.



Página Process Scheduler Request

La página Process Scheduler Request nos permite especificar variables, tales como dónde se ejecuta un proceso y en qué formato se genera la salida. Podemos configurar:

- Nombre del servidor
- Fecha de ejecución, tiempo de ejecución y zona horaria
- Recurrencia
- Tipo de salida
- Formato de salida
- Destino de salida
- Distribución

6.11.2 Modo manual: Línea de Comandos

Application Engine permite la ejecución de sus programas mediante línea de comandos. Esta es la sintaxis de la invocación:

```
psae
      -CT <dbtype>
      -CS <server>
      -CD <database name>
      -CO <oprid>
      -CP <oprpswd>
      -R  <run control id>
      -AI <program id>
      -I  <process instance>
      -DEBUG <Y|N>
      -DR <Y|N>
      -TRACE <value>
      -DBFLAGS <value>
      -TOOLSTRACESQL <value>
      -TOOLSTRACEPC <value>
      -OT <value>
      -OF <value>
      -FP<value>
```

Ejemplo:

```
set PS_SERVER_CFG=P:\CRMSYS\appserv\prcs\psinfsys\PSPRCS.CFG
set PS_SERVDIR=P:\tmp

P:\CRMDMO\bin\server\WINX86\psae -CT ORACLE -CD PSINFDMO -CO AFP00 -CP AFP00 -
R PA_EYC_RCI -AI PA_EYC -DEBUG N -TRACE 0 -DR Y -I 0
pause
```

La invocación por línea de comandos permite establecer si el programa es rearrancable (DR Y) o no, y el nivel de trazado que se desea para el programa (TRACE).

Para lanzar un programa desde la línea de comandos se requiere además de la existencia de algunas variables de entorno.

Dada la cantidad de parámetros que se introducen por línea de comandos, conviene encapsular la llamada en un shell script que contenga todos los parámetros y que si es necesario, establezca el valor de las variables de entorno necesarias.

Antes de ejecutar el proceso desde la línea de comando, es necesario solicitar la ejecución del programa desde la página de las PeopleTools, Process Request Page. Desde esta página se rellenan los campos de las tablas AEREQUESTTBL y AEREQUESTPARM. Esta página permite poner valores de inicialización a los registros de estado de un programa en la tabla AEREQUESTPARM.

Tras la ejecución del proceso, un campo de la tabla AEREQUESTTBL almacena el resultado de la última ejecución. Los valores que puede tomar son ‘P’ (pending), ‘E’ (errors) y ,’C’ (Completed). Si el planificador acepta un código de retorno para saber si el proceso fue mal

o no, se debería construir un pequeño script en SQL que consultara el estado final del proceso en base de datos. Se debe seleccionar por usuario y Run Control ID.

6.11.2.1 Trazas

En Application Engine, modificando los parámetros introducidos por línea de comando se puede cambiar nivel de trazas. También se pueden establecer a nivel global mediante la utilidad de configuración de PeopleSoft PSADMIN.

Las posibilidades de trazas que hay son:

- Trazas de cada uno de los pasos (steps) ejecutados.
- Trazas con cada una de las sentencias SQL, incluyendo parámetro de entrada.
- Trazas con estadísticas de ejecución de las sentencias SQL ejecutadas desde el Application Engine.
- Trazas con estadísticas de ejecución de código PeopleSoft.
- Trazas del optimizador de la base de datos.

Cada una de estas trazas se puede activar por separado en el parámetro –TRACE.

Esta es la lista de valores que admite. Para usar más de un valor a la vez, se debe sumar los valores deseados:

0	No trazar
1	Iniciar el trazado a nivel de paso de Application Engine.
2	Iniciar trazas SQL de Application Engine
4	Inicia trazado de la asignación de tablas temporales a los procesos.
128	Inicia el trazado de tiempos de ejecución. Esta funcionalidad ofrece un resumen que permite ver en qué se invirtió el tiempo durante la ejecución de proceso en Application Engine.
256	Similar al anterior, pero para PeopleCode. Initiates the PeopleCode Detail to the file for the Timings trace.
1024	Hace que las opciones 128 y 256 se vuelquen sobre tablas en lugar de volcarse sobre ficheros. Las tablas utilizadas son las tablas estándar

	PS_BAT_TIMINGS_LOG y PS_BAT_TIMINGS_DTL.
2048	Permite obtener trazas del optimizador de base de datos.
4096	Pide al optimizador que inserte trazas en la tabla Explain Plan, para dar información sobre el plan de ejecución de las sentencias SQL utilizadas.
8192	Establece una traza para los programas de transformación de Integration Broker.

6.11.2.2 Localización de ficheros de trazas

El lugar habitual donde se ubican las trazas de Application Engine para procesos batch está especificado en el campo Log/Output del fichero de configuración PS_SERVER_CFG. El nombre del fichero de trazas sigue la siguiente nomenclatura:

AE_<nombre_programa>_<Process_instance>.AET

6.11.3 Modo online: CallAppEngine

6.11.3.1 Sintaxis

La sintaxis para la llamada desde PeopleCode a un programa Application Engine es la siguiente:

```
CallAppEngine(applid [, statereclist, processinstance ]);
```

donde statereclist es una lista de objetos de tipo Record con la forma:

&staterecord1 [, &staterecord2]

Solo puede haber tantos objetos record en el statereclist como registros de estado haya en el programa Application Engine, el resto de objetos record son ignorados.

6.11.3.2 Descripción

CallAppEngine arranca el programa Application Engine nombrado applid. Esta es la forma de arrancar programas Application Engine de forma síncrona con PeopleCode desde una página.

Normalmente, no se ejecutan programas Application Engine de esta manera. La mayoría de los Application Engine se ejecutarán desde el Process Scheduler.

El processinstance nos permite especificar la instancia de proceso usada por el Application Engine en tiempo de ejecución.

Si al llamar a CallAppEngine se produce un fallo, todas las modificaciones en la base de datos se deshacen (roll back). Toda la información entrada por el usuario en el componente se pierde.

6.11.3.3 Parámetros

applid	Especifica el nombre del programa Application Engine que se va a arrancar
statereclist	Especifica un objeto record opcional que proporciona valores iniciales para un registro de estado.
processinstance	Especifica el process instance usado por el Application Engine en tiempo de ejecución.

6.11.3.4 Ejemplo

El siguiente ejemplo llama al programa Application Engine llamado MYAPPID, y le proporciona valores iniciales:

```
&REC = CreateRecord(RECORD.INIT_VALUES);

&REC.FIELD1.Value = "XYZ";
   /* set the initial value for INIT_VALUES.FIELD1 */

CallAppEngine("MYAPPID", &REC);
```

6.12 Descripción de programas con Application Engine

Para la compresión del manejo de todos los conceptos explicados anteriormente y entender la programación en PeopleSoft vamos a trabajar con una serie de programas de ejemplo de exportación/importación de datos, integrados sobre programas de Application Engine:

1) Exportación de una tabla a un fichero plano

2) Exportación de una tabla a un fichero plano utilizando Filelayout

3) Exportación de una tabla a un fichero plano utilizando Component Interface

4) Exportación de una tabla a un fichero plano utilizando Data Mover

5) Importación de fichero plano

6) Importación de fichero plano utilizando Filelayout

7) Importación de fichero plano utilizando Component Interface

8) Importación de fichero plano utilizando Data Mover

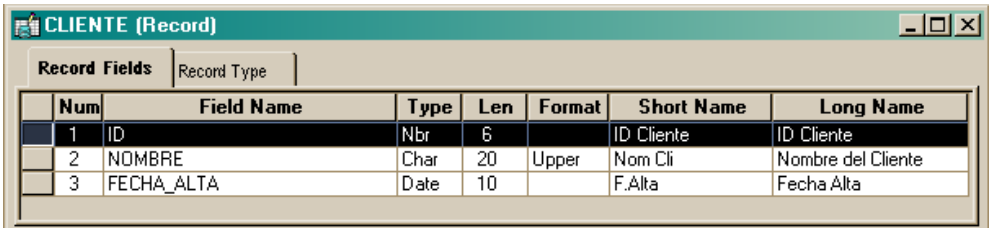
9) Importación de fichero plano utilizando Filelayout en modo re-arrancable

10) Copia de un fichero plano a otro fichero plano

11) Copia de una tabla a otra tabla

El formato de los ficheros planos se utilizarán serán ejemplos en CSV (valores separados por coma). Los campos están separados por comas y delimitados por dobles-comillas.

Para los siguientes programas de ejemplo utilizaremos la tabla PS_CLIENTE creada a partir del record CLIENTE, con el campo ID como campo clave.



Num	Field Name	Type	Len	Format	Short Name	Long Name
1	ID	Nbr	6		ID Cliente	ID Cliente
2	NOMBRE	Char	20	Upper	Nom Cli	Nombre del Cliente
3	FECHA_ALTA	Date	10		F.Alta	Fecha Alta

Record Cliente

Para el ejemplo de copia de una tabla a otra necesitaremos la tabla PS_CLIENTE_MOV, con la misma estructura que la tabla PS_CLIENTE.

En los ejemplos de programas de exportación e importación no se han implementado las características de rearrancabilidad estándar de programas Application Engine para simplificar la explicación de los mismos. Se ha incluido adicionalmente un programa rearrancable que muestra esta característica.

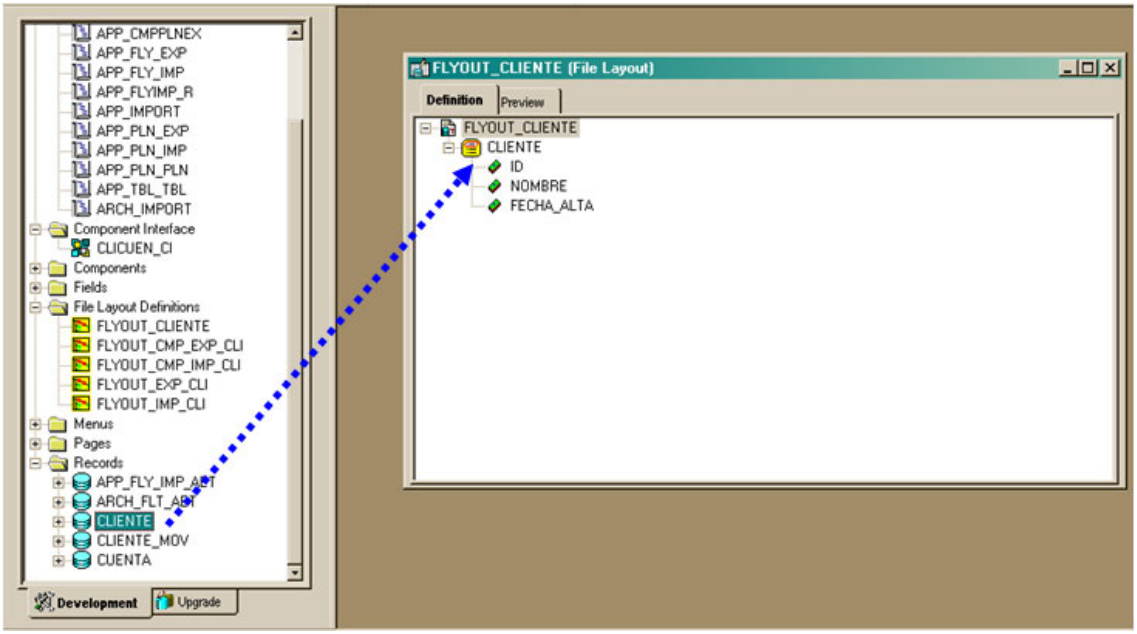
6.12.1 Creación de un FileLayout

Crearemos el objeto Filelayout FLYOUT_CLIENTE y lo utilizaremos en los ejemplos donde aplique.

Para ello en Application Designer elegiremos la opción File/New.../Filelayout. Lo guardamos con el nombre FLYOUT_CLIENTE (con la opción File/Save As...). Para incorporarlo al proyecto pulsamos F7.

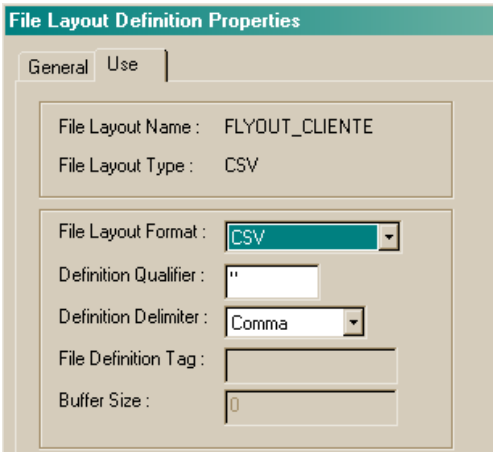
En dicho Filelayout enlazaremos el record CLIENTE arrastrando este objeto desde el panel de los objetos (situado a la izquierda) hasta el panel del Filelayout (situado en el centro).

El proceso se representa esquemáticamente en la siguiente figura:



Creación de Filelayout

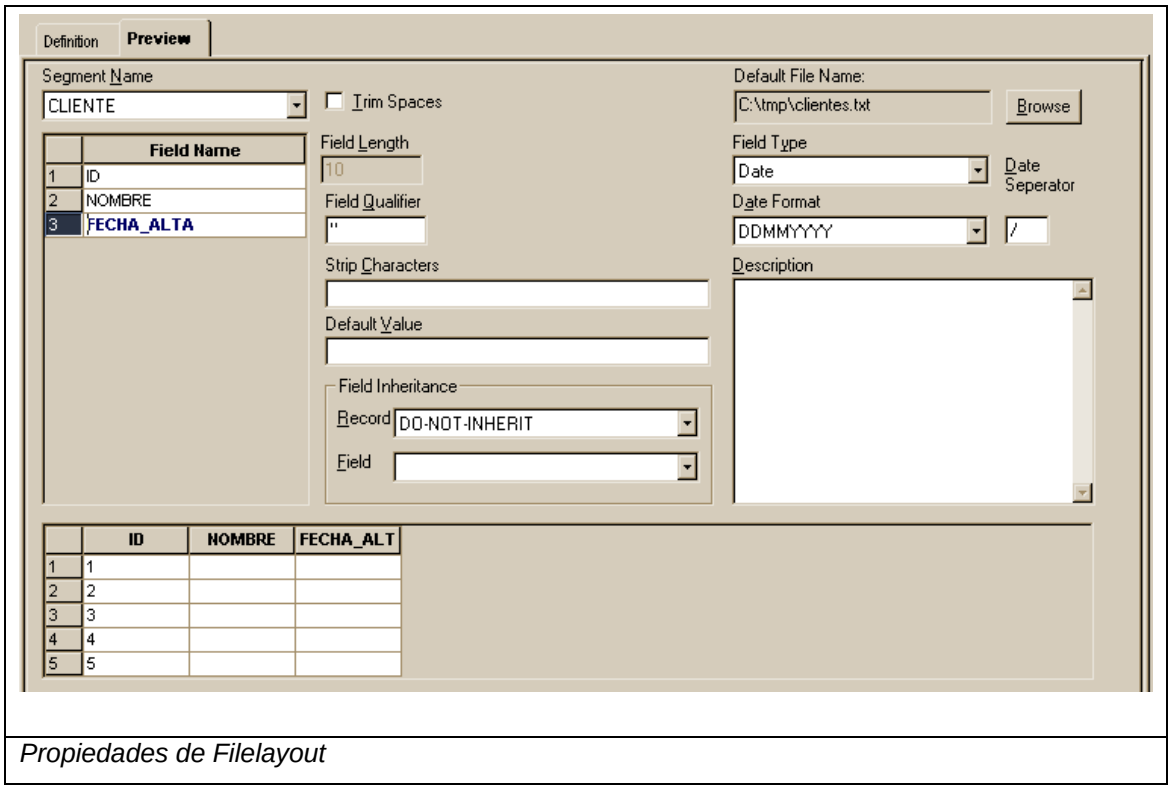
Una vez creado el Filelayout definimos las propiedades del Filelayout (tipo de formato del fichero y delimitador de campos principalmente) haciendo doble click sobre el mismo y eligiendo la pestaña “Use”:



Propiedades de Filelayout

Elegimos el formato del FileLayout como CSV (Valores Separados por Comas), el cualificador será el carácter de las dobles comillas y el delimitador de campos será la coma.

Después definimos las propiedades de los campos (Pestaña Preview del FileLayout):



Propiedades de Filelayout

En la figura observamos los valores a configurar:

1.- El campo cualificador en los tres campos será el carácter de las dobles comillas.

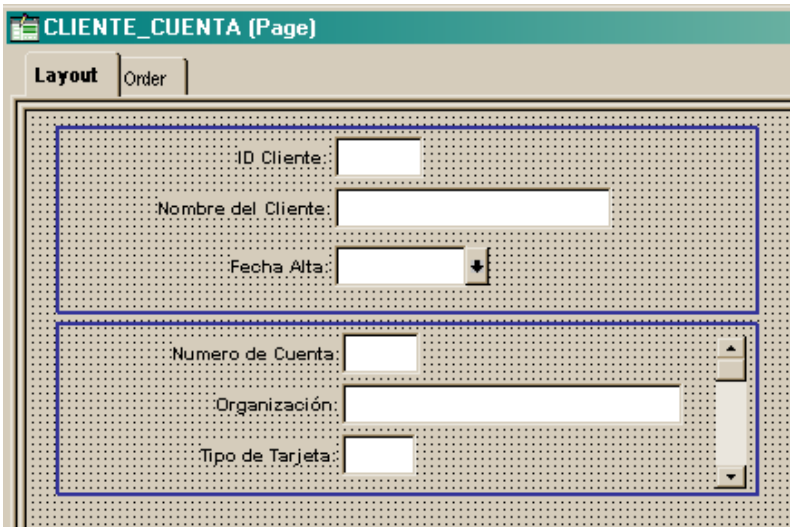
2.- El nombre de fichero por defecto tendrá la ruta: “c:\tmp\clientes.txt”

3.- El campo fecha tendrá el formato: DDMMYYYY separado por el carácter `/´.

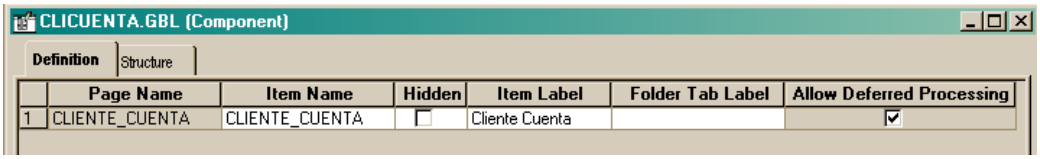
6.12.2 Creación de un Component Interface

Crearemos el objeto Component Interface CLICUEN_CI y lo utilizaremos en los ejemplos donde aplique. Para ello en Application Designer elegiremos la opción File/New.../Component Interface. A continuación introducimos el objeto componente CLICUENTA, previamente creado utilizando la página CLIENTE_CUENTA.

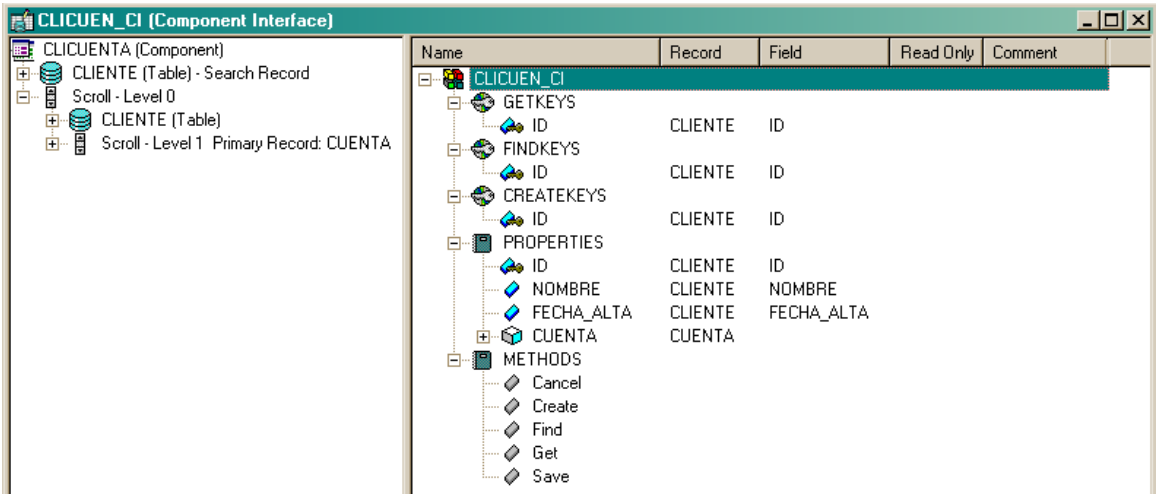
Lo guardamos con el nombre CLICUEN_CI, con la opción File/Save As... Para incorporarlo al proyecto pulsamos F7.



Página CLIENTE_CUENTA

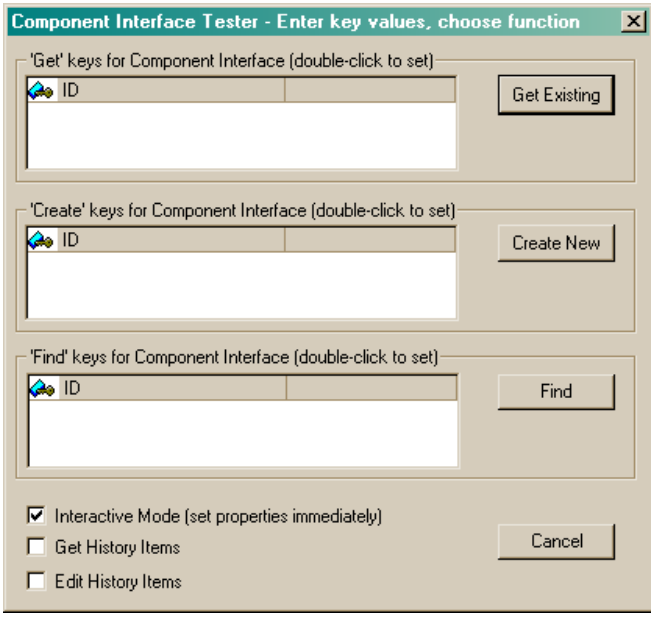


Componente CLICUENTA



Component Interface CLICUEN_CI

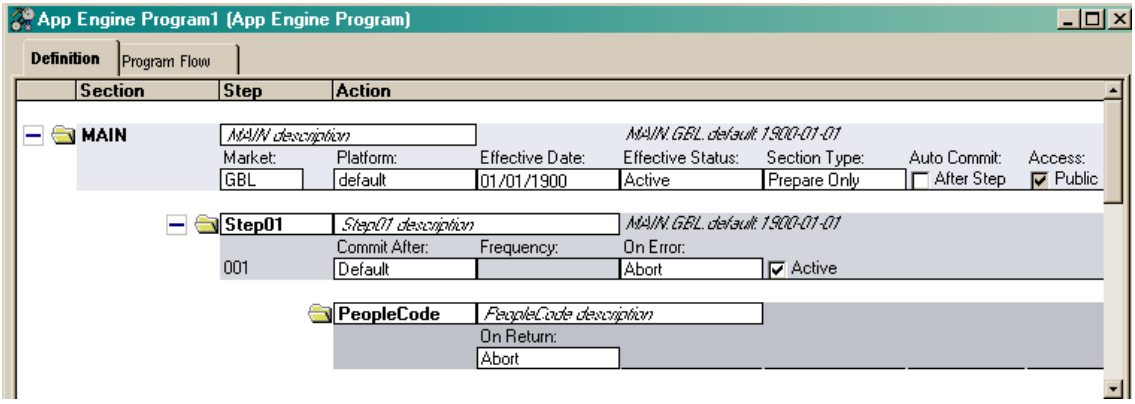
El Component Interface lo podemos probar a través de la herramienta del Application Designer: Tools/Test Component Interface. Esta ventana de testeo se muestra en la siguiente figura:



Component Interface Tester

Una vez probado podemos generar el código para utilizar los diferentes métodos del Component Interface.

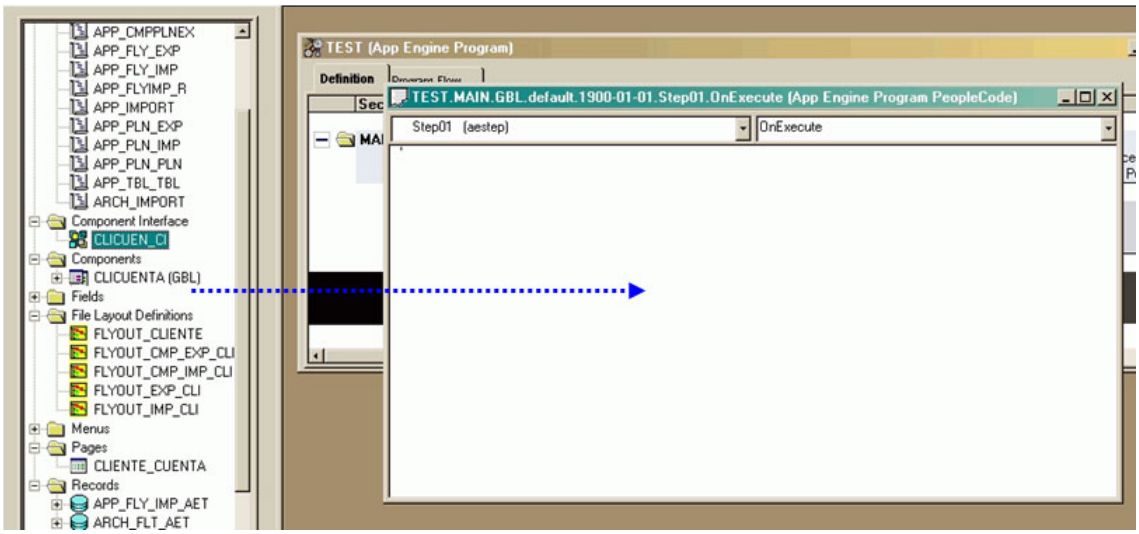
Para hacer esto creamos un nuevo programa Application Engine (File/New.../App Engine Program). Para ello seleccionamos un paso dentro de una sección, incluimos una acción (Insert/Action) y elegimos PeopleCode como tipo de acción. La estructura del programa quedaría así:



Application Engine

Una vez que grabemos el programa (File/Save As...) hacemos click en botón derecho del ratón y seleccionamos: View PeopleCode.

Después en el panel de los objetos (Project Workspace) situado a la izquierda seleccionamos el Component Interface y lo arrastramos al panel del PeopleCode que acabamos de abrir y soltamos. Entonces aparecerá en dicho panel el código PeopleCode para acceder a los métodos del Component Interface.



Component Interface PeopleCode

Este código generado es una plantilla y es usado como una ayuda para el desarrollador, que tendrá que reemplazar las marcas '<*>' o los valores por defecto con referencias a variables PeopleCode y/o campos y records. También tendrá que descomentar los métodos que quiera utilizar (comentados con la palabra clave 'rem' o las marcas /* <código> */.

```
Local ApiObject &oSession;
Local ApiObject &oClicuenCi;
Local ApiObject &oCuentaCollection;
Local ApiObject &oCuenta;
Local ApiObject &oPSMessageCollection;
Local ApiObject &oPSMessage;
Local File &LogFile;
Local number &i;
Local String &strErrMsgSetNum, &strErrMsgNum, &strErrMsgText, &strErrType;

Function errorHandler()
       &oPSMessageCollection = &oSession.PSMessages;
       For &i = 1 To &oPSMessageCollection.Count
              &oPSMessage = &oPSMessageCollection.Item(&i);
              &strErrMsgSetNum = &oPSMessage.MessageSetNumber;
```

```
            &strErrMsgNum = &oPSMessage.MessageNumber;
            &strErrMsgText = &oPSMessage.Text;
            &LogFile.WriteLine(&strErrType | " (" | &strErrMsgSetNum | "," |
&strErrMsgNum | ") - " | &strErrMsgText);
      End-For;
      rem ***** Delete the Messages from the collection *****;
      &oPSMessageCollection.DeleteAll();
End-Function;

rem ***** Set the Log File *****;
&LogFile = GetFile("C:\temp\CLICUEN_CI.log", "w", "a", %FilePath_Absolute);
&LogFile.WriteLine("Begin");
rem ***** Get current PeopleSoft Session *****;
&oSession = %Session;

rem ***** Set the PeopleSoft Session Error Message Mode *****;
rem ***** None *****;
rem &oSession.PSMessagesMode = 0;
rem ***** PSMessage Collection only (default) *****;
&oSession.PSMessagesMode = 1;
rem ***** Message Box only *****;
rem &oSession.PSMessagesMode = 2;
rem ***** Both collection and message box *****;
rem &oSession.PSMessagesMode = 3;

rem ***** Get the Component Interface *****;
&oClicuenCi = &oSession.GetCompIntfc(CompIntfc.CLICUEN_CI);
If &oClicuenCi = Null Then
      errorHandler();
      Exit;
End-If;

rem ***** Set the Component Interface Mode *****;
&oClicuenCi.InteractiveMode = False;
&oClicuenCi.GetHistoryItems = True;
&oClicuenCi.EditHistoryItems = False;

rem ***** Set Component Interface Get/Create Keys *****;
&oClicuenCi.ID = <*>;

rem ***** Execute Get *****;
If Not &oClicuenCi.Get() Then
      rem ***** No rows exist for the specified keys. Failed to get the
Component Interface. *****;
      errorHandler();
      Exit;
End-If;

rem ***** Execute Create ******;
/*If Not &oClicuenCi.Create() Then
      rem ***** Unable to Create Component Interface for the Add keys
provided. *****;
      errorHandler();
      Exit;
End-If;*/


rem ***** Begin: Get/Set Component Interface Properties *****;
rem ***** Get/Set Level 0 Field Properties *****;
rem <*> = &oClicuenCi.ID;
rem &oClicuenCi.ID = <*>;
rem <*> = &oClicuenCi.NOMBRE;
rem &oClicuenCi.NOMBRE = <*>;
rem <*> = &oClicuenCi.FECHA_ALTA;
rem &oClicuenCi.FECHA_ALTA = <*>;
```

```
rem ***** Set/Get CUENTA Collection Field Properties -- Parent: PS_ROOT
Collection *****;
rem &oCuentaCollection = &oClicuenCi.CUENTA;
rem For &<*> = 1 To &oCuentaCollection.Count
rem &oCuenta = &oCuentaCollection.Item(&<*>);
rem <*> = &oCuenta.NUM_CUENTA;
rem &oCuenta.NUM_CUENTA = <*>;
rem <*> = &oCuenta.ORGANIZACION;
rem &oCuenta.ORGANIZACION = <*>;
rem <*> = &oCuenta.TIPO_TAR;
rem &oCuenta.TIPO_TAR = <*>;
rem End-For;

rem ***** End: Get/Set Component Interface Properties *****;
rem ***** Execute Save *****;
/*If Not &oClicuenCi.Save() Then
      errorHandler();
      Exit;
End-If;*/

rem ***** Execute Cancel *****;
/*If Not &oClicuenCi.Cancel() Then
      errorHandler();
      Exit;
End-If;*/

&LogFile.WriteLine("End");
&LogFile.Close();
```

6.12.3 Descripción de programas: Exportación

6.12.3.1 Tabla → Fichero Plano

APP_PLN_EXP: Programa que lee de una tabla y escribe en un fichero. Este programa es re-arrancable solamente en el sentido de que si en algún momento de la ejecución el proceso se interrumpe, aunque el fichero plano creado no contenga todas las filas de la tabla bastaría con volver a ejecutar el programa y reescribiría el contenido del fichero.

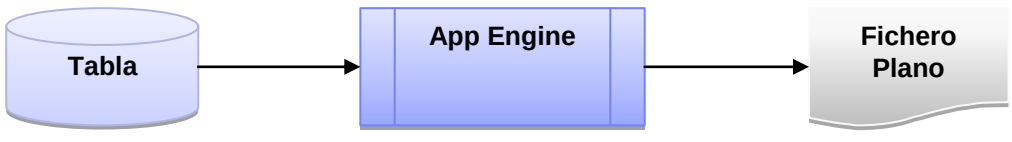
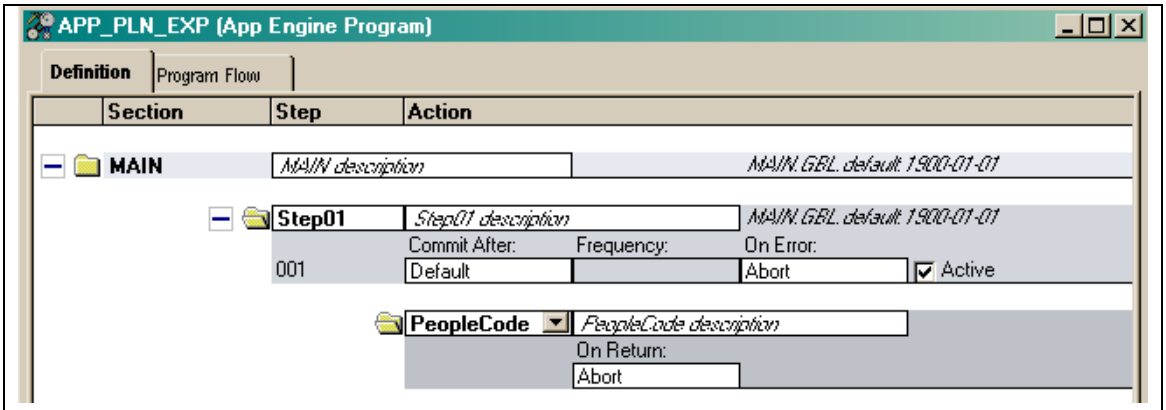


Tabla – App Engine -- Fich

La tabla PS_CLIENTE tendrá que contener los registros que queremos exportar. El formateo de los campo se hace por código.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Primero abrimos el fichero clientes.txt en modo escritura (si ya existe se vuelve a crear perdiendo los datos). Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas). Creamos un objeto record con la estructura de la tabla PS_CLIENTE, después realizamos una sentencia “SELECT * FROM PS_CLIENTE” para recuperar todos los registros de la tabla que vamos leyendo fila-a-fila y en cada fila, concatenamos todos los campos de cada registro en una variable de tipo String y cuando acabemos con cada fila la escribimos en el fichero de texto. Si el campo es el último (Fecha_Alta) lo formateamos para que se muestre en el fichero de texto como “dd/mm/yyyy”. Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros exportados y otro con la fecha y hora de fin de ejecución.

```
Global number &RowsExported;
Local File &EXPFILE;
Local Record &REC1;

&EXPFILE = GetFile("C:\tmp\clientes.txt", "W", "A", %FilePath_Absolute);

If &EXPFILE.IsOpen Then

   MessageBox(0, "Exportacion", 20002, 1, "Message", %Datetime);
```

```
   &REC1 = CreateRecord(Record.CLIENTE);
   &FIELDCOUNT = &REC1.FieldCount;
   &SQL1 = CreateSQL("%Selectall(:1)", &REC1);
   While &SQL1.Fetch(&REC1)

      &STRING1 = "";

      For &I = 1 To &FIELDCOUNT

         &FLD1 = &REC1.GetField(&I).Value;

         If &I <> &FIELDCOUNT Then

            &STRING1 = &STRING1 | &FLD1;
            &STRING1 = &STRING1 | ",";
         Else

            /**** Campo Fecha : 2015-01-01 => 01/01/2015 */

            &CANIO = Substring(&FLD1, 1, 4);
            &CMES = Right(&FLD1, 2);
            &CDIA = Substring(&FLD1, 9, 2);

            &STRING1 = &STRING1 | &CDIA | "/" | &CMES | "/" | &CANIO;
         End-If;

      End-For;
      &EXPFILE.WriteLine(&STRING1);
      &RowsExported = &RowsExported + 1;
   End-While; /* end &SQL1.Fetch */


   MessageBox(0, "Exportacion", 20003, 1, "Message", %Datetime);

   MessageBox(0, "Exportacion", 20000, 1, "Message", &RowsExported);

   &EXPFILE.Close();
End-If;
```

6.12.3.2 Tabla → Fichero Plano con Filelayout

APP_FLY_EXP: Programa que lee de una tabla y escribe en un fichero (Utilizando Filelayout). Este programa es re-arrancable solamente en el sentido de que si en algún momento de la ejecución el proceso se interrumpe, aunque el fichero plano creado no contenga todas las filas de la tabla bastaría con volver a ejecutar el programa y reescribiría el contenido del fichero.

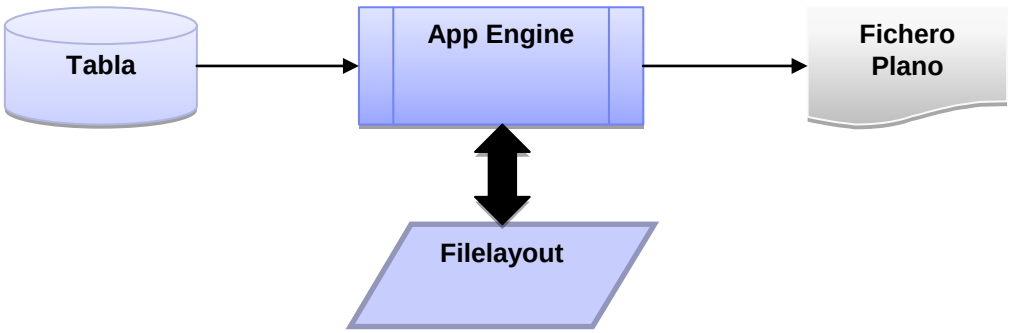
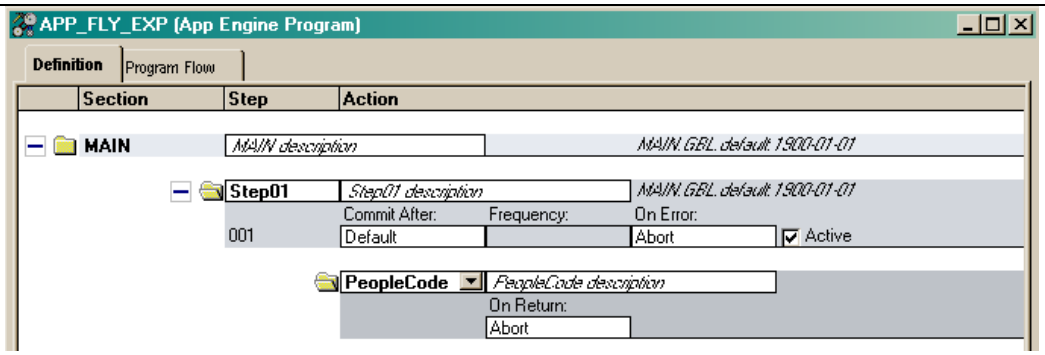


Tabla a Fichero con Filelayout

La tabla PS_CLIENTE tendrá que contener los registros que queremos exportar. El formateo de los campos no se hace por código sino a través del FileLayout.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Primero abrimos el fichero clientes.txt en modo escritura (si ya existe se vuelve a crear perdiendo los datos). Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas). Después asignamos el Filelayout FLYOUT_CLIENTE con el fichero creado.

Creamos un objeto record con la estructura de la tabla PS_CLIENTE, después realizamos una sentencia “SELECT * FROM PS_CLIENTE” para recuperar todos los registros de la tabla que vamos leyendo fila-a-fila y en cada fila, copiamos el registro leído a una estructura RowSet ligada al FileLayout, y con el método WriteRowSet escribimos los datos en el fichero. Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros exportados y otro con la fecha y hora de fin de ejecución.

De esta manera se consigue una mayor rapidez a la hora de escribir los registros.

```
Global number &RowsExported;
Local Record &REC1;
Local File &File;
Local Rowset &RS1;
Local SQL &SQL1;


&File = GetFile("C:\tmp\clientes.txt", "W", "A", %FilePath_Absolute);

If &File.IsOpen Then

   MessageBox(0, "Exportacion", 20002, 1, "Message", %Datetime);

   If &File.SetFileLayout(FileLayout.FLYOUT_CLIENTE) Then

      &REC1 = CreateRecord(Record.CLIENTE);
      &RS1 = &File.CreateRowset();
      &SQL1 = CreateSQL("%selectall(:1)", &REC1);

      While &SQL1.Fetch(&REC1)

         &REC1.CopyFieldsTo(&RS1.GetRow(1).CLIENTE);

         &File.WriteRowset(&RS1, True);

         &RowsExported = &RowsExported + 1;

      End-While;

   End-If;

   MessageBox(0, "Exportacion", 20003, 1, "Message", %Datetime);

   MessageBox(0, "Exportacion", 20000, 1, "Message", &RowsExported);

End-If;

&File.Close();
```

6.12.3.3 Tabla → Fichero Plano Con Filelayout + Component Interface

- APP_CMP_EXP: Este programa lee datos de una tabla utilizando un FileLayout y un Component Interface. Este programa es re-arrancable solamente en el sentido de que si en algún momento de la ejecución el proceso se interrumpe, aunque el fichero plano creado no contenga todas las filas de la tabla bastaría con volver a ejecutar el programa y reescribiría el contenido del fichero.

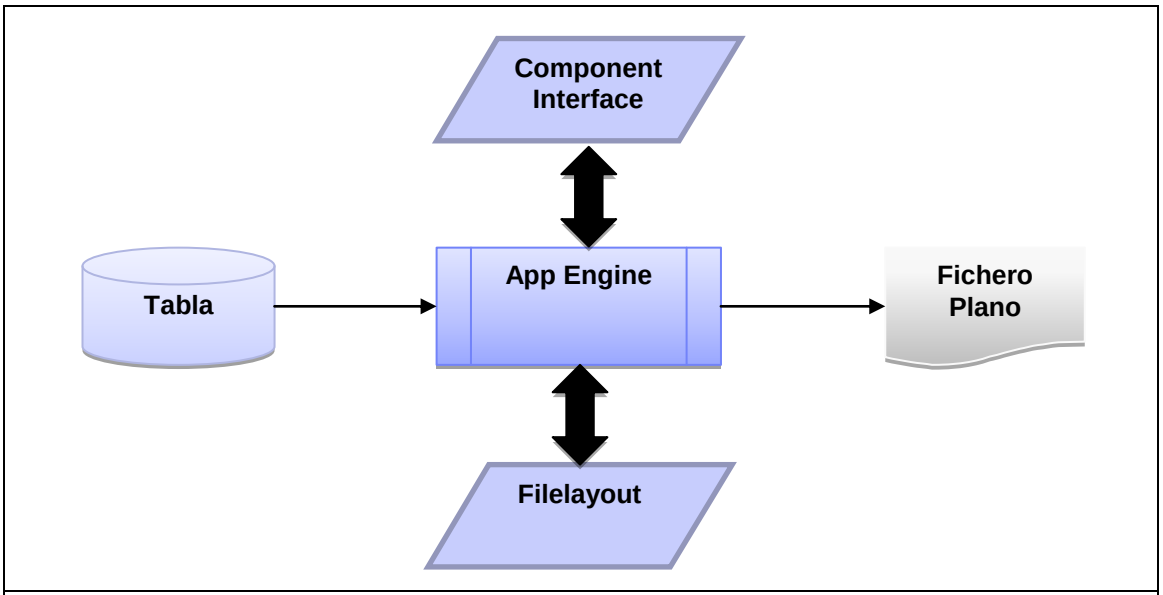
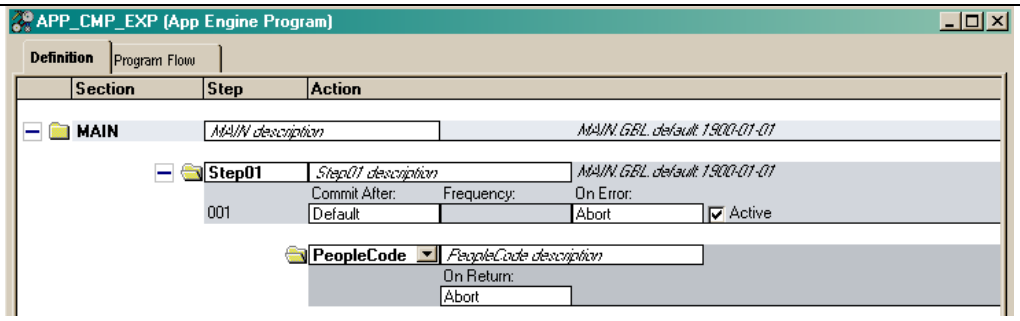


Tabla a Fichero con Filelayout + Comp Interface

La tabla PS_CLIENTE tendrá que contener los registros que queremos exportar. Este programa es muy lento porque vamos recuperando cada registro de la tabla con el método Get.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Utilizaremos la plantilla que genera el Component Interface, el código marcado es el código modificado para realizar la exportación.

Primero abrimos el fichero clientes.txt en modo escritura (si ya existe se vuelve a crear perdiendo los datos). Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas). Después asignamos el Filelayout FLYOUT_CLIENTE con el fichero creado.

Después realizamos dos consultas sobre la tabla PS_CLIENTE, una para saber el ID mínimo y otra para saber el máximo. Luego hacemos un bucle desde el mínimo al máximo y por cada iteración recuperamos los datos con el método Get y el campo clave, los guardamos en el registro y los copiamos al RowSet para después escribirlo en el fichero con el método WriteRowSet.

Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros exportados y otro con la fecha y hora de fin de ejecución.

```
Global number &RowsExported;
Local Record &REC1;
Local File &File;
Local Rowset &RS1;
Local SQL &SQL1;
Local SQL &SQL_MIN;
Local SQL &SQL_MAX;

Local ApiObject &oSession;
Local ApiObject &oClicuenCi;
Local ApiObject &oCuentaCollection;
Local ApiObject &oCuenta;
Local ApiObject &oPSMessageCollection;
Local ApiObject &oPSMessage;
Local File &LogFile;
```

```
Local number &i;
Local string &strErrMsgSetNum, &strErrMsgNum, &strErrMsgText, &strErrType;

Function errorHandler()
   &oPSMessageCollection = &oSession.PSMessages;
   For &i = 1 To &oPSMessageCollection.Count
      &oPSMessage = &oPSMessageCollection.Item(&i);
      &strErrMsgSetNum = &oPSMessage.MessageSetNumber;
      &strErrMsgNum = &oPSMessage.MessageNumber;
      &strErrMsgText = &oPSMessage.Text;
      &LogFile.WriteLine(&strErrType | " (" | &strErrMsgSetNum | "," |
&strErrMsgNum | ") - " | &strErrMsgText);
   End-For;
   rem ***** Delete the Messages from the collection *****;
   &oPSMessageCollection.DeleteAll();
End-Function;

&File = GetFile("C:\tmp\clientes.txt", "W", "A", %FilePath_Absolute);

If &File.IsOpen Then

   MessageBox(0, "Exportacion", 20002, 1, "Message", %Datetime);

   If &File.SetFileLayout(FileLayout.FLYOUT_CLIENTE) Then

      &SQL_MIN = CreateSQL("select MIN(ID) from PS_CLIENTE");
      &SQL_MAX = CreateSQL("select MAX(ID) from PS_CLIENTE");

      If &SQL_MIN.Fetch(&NUM_MIN) Then
         MessageBox(0, "Exportacion", 20004, 1, "Message", "MIN: " |
&NUM_MIN);
      End-If;
      If &SQL_MAX.Fetch(&NUM_MAX) Then
         MessageBox(0, "Exportacion", 20004, 1, "Message", "MAX: " |
&NUM_MAX);
      End-If;


      rem ***** Set the Log File *****;
      &LogFile = GetFile("C:\temp\CLICUEN_CI.log", "w", "a",
%FilePath_Absolute);
      &LogFile.WriteLine("Begin");


      rem ***** Get current PeopleSoft Session *****;
      &oSession = %Session;

      rem ***** Set the PeopleSoft Session Error Message Mode *****;
      rem ***** PSMessage Collection only (default) *****;
      &oSession.PSMessagesMode = 1;


      For &I_INDEX = &NUM_MIN To &NUM_MAX

         rem ***** Get the Component Interface *****;
         &oClicuenCi = &oSession.GetCompIntfc(CompIntfc.CLICUEN_CI);
         If &oClicuenCi = Null Then
            errorHandler();
            Exit;
         End-If;

         rem ***** Set the Component Interface Mode *****;
         &oClicuenCi.InteractiveMode = False;
         &oClicuenCi.GetHistoryItems = True;
         &oClicuenCi.EditHistoryItems = False;

         rem ***** Set Component Interface Get/Create Keys *****;
         &oClicuenCi.ID = &I_INDEX;
```

```
         rem ***** Execute Get *****;
         If Not &oClicuenCi.Get() Then
            rem ***** No rows exist for the specified keys. Failed to get the
Component Interface.*****;
            errorHandler();
            Exit;
         End-If;

         &REC1 = CreateRecord(Record.CLIENTE);

         rem ***** Begin: Get/Set Component Interface Properties *****;
         rem ***** Get/Set Level 0 Field Properties *****;

         &REC1.ID.Value = &oClicuenCi.ID;
         &REC1.NOMBRE.Value = &oClicuenCi.NOMBRE;
         &REC1.FECHA_ALTA.Value = &oClicuenCi.FECHA_ALTA;

         &RS1 = &File.CreateRowset();


         &REC1.CopyFieldsTo(&RS1.GetRow(1).CLIENTE);

         &File.WriteRowset(&RS1, True);

         &RowsExported = &RowsExported + 1;


      End-For;

   End-If;

   MessageBox(0, "Exportacion", 20003, 1, "Message", %Datetime);

   MessageBox(0, "Exportacion", 20000, 1, "Message", &RowsExported);

End-If;

&LogFile.Close();

&File.Close();
```

6.12.3.4 Tabla → Fichero Plano Con Datamover

- CLIENTES_EXPORT.dms: Programa que lee de una tabla y escribe el resultado en un fichero csv utilizando Datamover.

Tabla a fichero con DataMover

La tabla PS_CLIENTE tendrá que contener los registros que queremos exportar.

El fichero plano creado por DataMover (clientes.dat) tiene que ser un fichero CSV, este método es muy rápido, pero el fichero creado solo es válido para PeopleSoft ya que está creado con un formato específico.

6.12.3.5 Script Data Mover

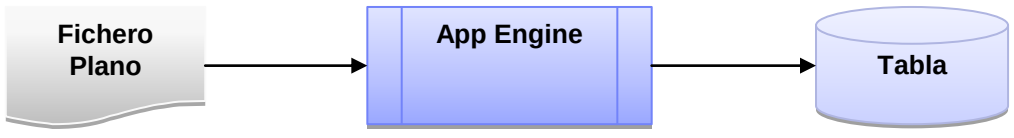
Con Set Log indicamos la ruta y el fichero donde estará el fichero log y con Set Output indicamos ruta y fichero donde se exportarán los datos de la tabla PS_CLIENTE. Para ejecutar este script necesitamos la aplicación DataMover que se incluye en el paquete comercial de PeopleSoft.

```
SET NO TRACE;
SET LOG C:\TEMP\EXPPROJ.LOG;
SET OUTPUT C:\TEMP\CLIENTES.DAT;

EXPORT PS_CLIENTE;
```

6.12.4 Descripción de programas: Importación

6.12.4.1 Fichero Plano → Tabla

- APP_PLN_IMP: Programa que lee de un fichero y escribe en una tabla. Este programa utilizado en modo multi-usuario no es re-arrancable, si en algún momento de la ejecución el proceso se interrumpe podría haber inconsistencia de datos.



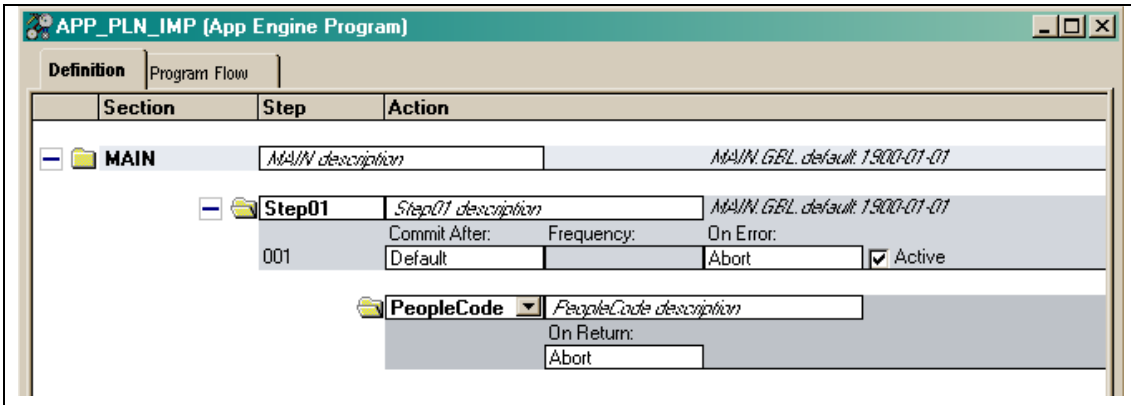
Fich – App Engine -- Tabla

La tabla PS_CLIENTE no deberá contener los registros que queremos importar para que no se produzcan errores de duplicidad de claves. Es aconsejable que esté vacía. El formateo de los campos se hace por código.

Este programa al utilizar el método BulkMode es bastante rápido porque los inserts en la tabla se gestionan por bloques.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Primero abrimos el fichero clientes.txt en modo lectura. Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas).

Creamos un objeto record con la estructura de la tabla PS_CLIENTE, después vamos leyendo todos los registros del fichero y en cada línea, con la sentencia Split, dividimos la línea por el carácter "," que es el separador de campos, el resultado se guarda en un array.

El array con los datos de la línea lo vamos leyendo con la sentencia Shift, que va recuperando cada dato y rotando para leer los siguientes. Cuando recuperamos el tercer campo, que es la fecha de alta, lo formateamos como "yyyy-mm-dd" y lo convertimos a tipo Date. Los campos los introducimos en el record CLIENTE y hacemos el insert con este objeto record.

Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros importados y otro con la fecha y hora de fin de ejecución.

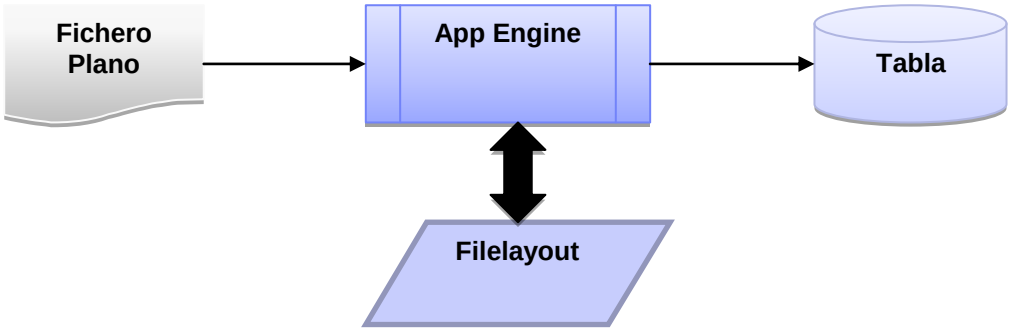
```
Local File &IMPFILE;
Local Record &REC1;
Local SQL &SQL1;
Global number &RowsImported;

&IMPFILE = GetFile("C:\tmp\clientes.txt", "R", "A", %FilePath_Absolute);
```

```
&SQL1 = CreateSQL("%Insert(:1)");


/*** BULK ****/
&SQL1.BulkMode = True;


If &IMPFILE.IsOpen Then

   MessageBox(0, "Importacion", 20002, 1, "Message", %Datetime);

   &REC1 = CreateRecord(Record.CLIENTE);

   While &IMPFILE.ReadLine(&LINEA)

      &CAMPOS = Split(&LINEA, ",");

      &LEN_ARR = &CAMPOS.Len;

      For &I = 1 To &LEN_ARR;

         &ITEM = &CAMPOS.Shift();

         Evaluate &I
         When 1
            &REC1.ID.Value = Value(&ITEM);
         When 2
            &REC1.NOMBRE.Value = &ITEM;
         When 3
            &ANIO = Right(&ITEM, 4);
            &MES = Substring(&ITEM, 4, 2);
            &DIA = Left(&ITEM, 2);

            &NEWDATE = &ANIO | "-" | &MES | "-" | &DIA;

            &REC1.FECHA_ALTA.Value = Date(&NEWDATE);
         End-Evaluate;
      End-For;

      /* &REC1.Insert(); */

      &TODO_OK = &SQL1.Execute(&REC1);

      If Not &TODO_OK Then
         MessageBox(0, "Importacion", 20004, 1, "Message", "ERROR !");

         Evaluate &SQL1.Status
         When = %SQLStatus_OK
            MessageBox(0, "Importacion", 20004, 1, "Message", "It worked.");
            Exit;
         When = %SQLStatus_NotFound
            MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key were
not found.");
            Exit;
         When = %SQLStatus_Duplicate
            MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key were
already there.");
            Exit;
         End-Evaluate;
      End-If;

      &RowsImported = &RowsImported + 1;

   End-While;

   MessageBox(0, "Importacion", 20003, 1, "Message", %Datetime);

   MessageBox(0, "Importacion", 20001, 1, "Message", &RowsImported);
```

```
   &IMPFILE.Close();

End-If;
```

6.12.4.2 Fichero Plano Con Filelayout → Tabla

- APP_FLY_IMP: Programa que lee de un fichero y escribe en una tabla (Utilizando Filelayout). Este programa utilizado en modo multi-usuario no es re-arrancable, si en algún momento de la ejecución el proceso se interrumpe podría haber inconsistencia de datos.



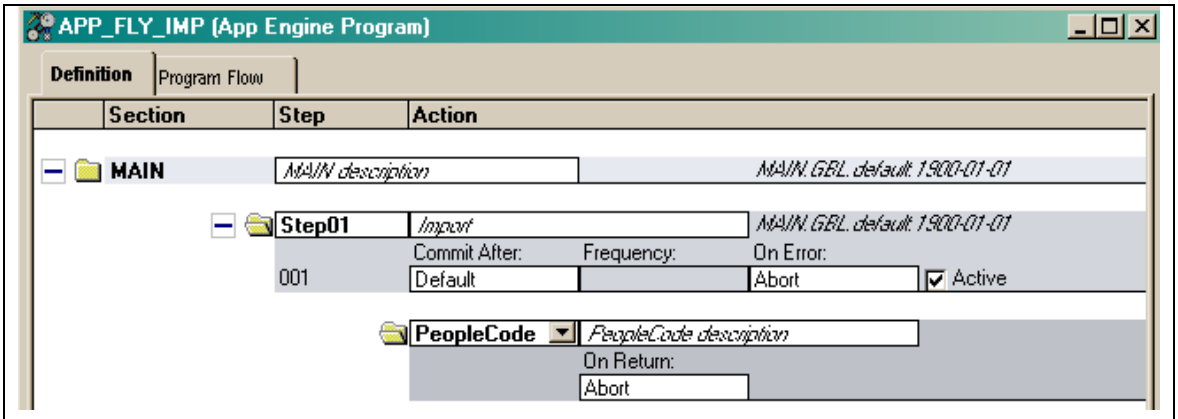
Fichero a Tabla con Filelayout

La tabla PS_CLIENTE no deberá contener los registros que queremos importar para que no se produzcan errores de duplicidad de claves. Es aconsejable que esté vacía. El formateo de los campos no se hace por código sino a través del Filelayout.

Este programa al utilizar el método BulkMode es bastante rápido porque los inserts en la tabla se gestionan por bloques.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Primero abrimos el fichero clientes.txt en modo lectura. Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas).

Enlazamos el fichero con el Filelayout FLYOUT_CLIENTE, recuperamos la estructura de la tabla con el método ReadRowSet en un objeto y lo vamos leyendo, por cada registro leído lo copiamos en un objeto record y ejecutamos el insert.

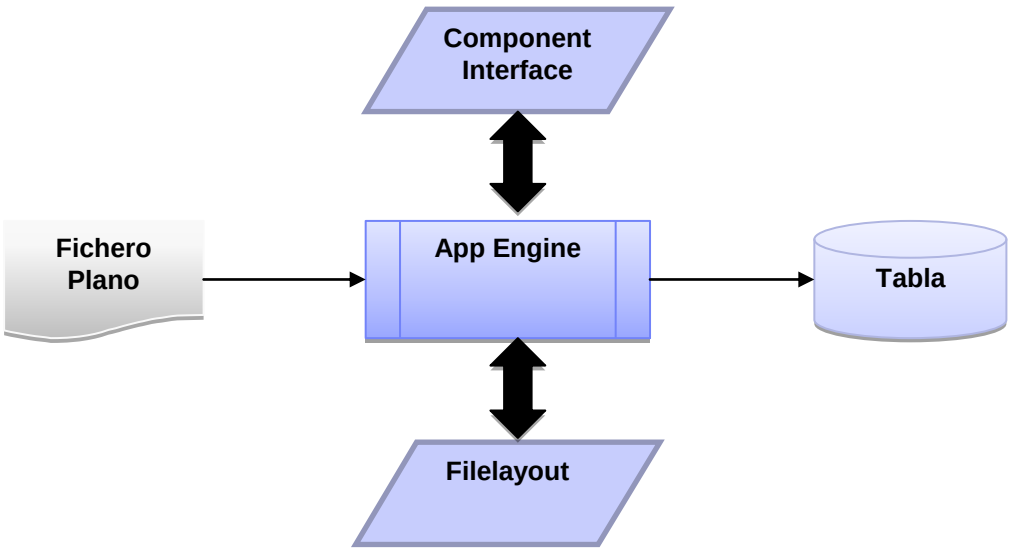
Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros importados y otro con la fecha y hora de fin de ejecución.

```
Global number &RowsImported;
Local Record &REC1;
Local File &File;
Local Rowset &RS1;
Local SQL &SQL1;

&File = GetFile("C:\tmp\clientes.txt", "R", "A", %FilePath_Absolute);
&REC1 = CreateRecord(Record.CLIENTE);
&SQL1 = CreateSQL("%Insert(:1)");

/****** BULKMODE  **********/
&SQL1.BulkMode = True;

If &File.IsOpen Then

   MessageBox(0, "Importacion", 20002, 1, "Message", %Datetime);

   If &File.SetFileLayout(FileLayout.FLYOUT_CLIENTE) Then

      &RS1 = &File.ReadRowset();


      While &RS1 <> Null
```

```
         &RS1.GetRow(1).CLIENTE.CopyFieldsTo(&REC1);
         &TODO_OK = &SQL1.Execute(&REC1);

         If Not &TODO_OK Then
            MessageBox(0, "Importacion", 20004, 1, "Message", "ERROR !");

            Evaluate &SQL1.Status
            When = %SQLStatus_OK
               MessageBox(0, "Importacion", 20004, 1, "Message", "It
worked.");
               Exit;
            When = %SQLStatus_NotFound
               MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key
were not found.");
               Exit;
            When = %SQLStatus_Duplicate
               MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key
were already there.");
               Exit;
            End-Evaluate;

         End-If;

         &RowsImported = &RowsImported + 1;

         &RS1 = &File.ReadRowset();

      End-While;

   End-If;

   MessageBox(0, "Importacion", 20002, 1, "Message", %Datetime);
   MessageBox(0, "Importacion", 20001, 1, "Message", &RowsImported);

   &File.Close();

   &SQL1.Close();

End-If;
```

6.12.4.3 Fichero Plano Con Component Interface → Tabla

- APP_CMP_IMP: Programa que lee de un fichero y escribe en una tabla (utilizando Filelayout y Component Interface). Este programa utilizado en modo multi-usuario no es re-arrancable, si en algún momento de la ejecución el proceso se interrumpe podría haber inconsistencia de datos.



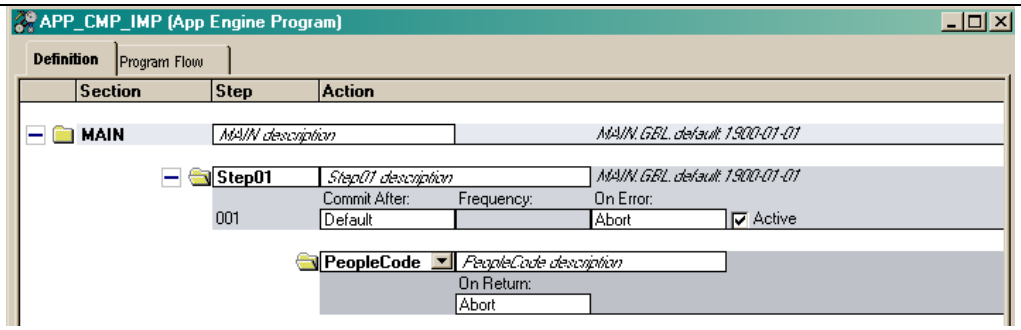
Fichero a Tabla con Filelayout + Comp Interface

La tabla PS_CLIENTE no deberá contener los registros que queremos importar para que no se produzcan errores de duplicidad de claves. Es aconsejable que esté vacía. Este programa es muy lento porque por cada registro leído del Filelayout ejecuta un método Create y un método Save para guardar la información en la base de datos.

El método Save no hace commit en la base de datos, el commit se hace cuando finaliza el paso Step01.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura del App Engine

2.- CÓDIGO DEL PROGRAMA:

Utilizaremos la plantilla que genera el Component Interface, el código marcado es el código que hemos incluido para realizar la importación.

Primero abrimos el fichero clientes.txt en modo lectura. Si la apertura del fichero ha sido correcta mostramos un mensaje con fecha y hora de comienzo (para estadísticas).

Enlazamos el fichero con el Filelayout FLYOUT_CLIENTE, recuperamos la estructura de la tabla con el método ReadRowSet en un objeto y lo vamos leyendo, por cada registro leído lo recuperamos una instancia del Component Interface, llamamos al método Create, rellenamos las propiedades y atributos (campos de la tabla) y ejecutamos el método Save.

Cuando acabamos de leer toda la tabla mostramos un mensaje con el número de registros importados y otro con la fecha y hora de fin de ejecución.

```
Global number &RowsImported;
Local Record &REC1;
Local File &File;
Local Rowset &RS1;
Local SQL &SQL1;

Local ApiObject &oSession;
Local ApiObject &oClicuenCi;
Local ApiObject &oCuentaCollection;
Local ApiObject &oCuenta;
Local ApiObject &oPSMessageCollection;
Local ApiObject &oPSMessage;
Local File &LogFile;
Local number &i;
Local string &strErrMsgSetNum, &strErrMsgNum, &strErrMsgText, &strErrType;

Function errorHandler()
   &oPSMessageCollection = &oSession.PSMessages;
   For &i = 1 To &oPSMessageCollection.Count
      &oPSMessage = &oPSMessageCollection.Item(&i);
      &strErrMsgSetNum = &oPSMessage.MessageSetNumber;
      &strErrMsgNum = &oPSMessage.MessageNumber;
      &strErrMsgText = &oPSMessage.Text;
      &LogFile.WriteLine(&strErrType | " (" | &strErrMsgSetNum | "," |
&strErrMsgNum | ") - " | &strErrMsgText);
   End-For;
   rem ***** Delete the Messages from the collection *****;
   &oPSMessageCollection.DeleteAll();
End-Function;

&File = GetFile("C:\tmp\clientes.txt", "R", "A", %FilePath_Absolute);
&REC1 = CreateRecord(Record.CLIENTE);
&SQL1 = CreateSQL("%Insert(:1)");

If &File.IsOpen Then

   rem ***** Set the Log File *****;
   &LogFile = GetFile("C:\temp\CLICUEN_CI.log", "w", "a", %FilePath_Absolute);
   &LogFile.WriteLine("Begin");
```

```
   MessageBox(0, "Importacion", 20002, 1, "Message", %Datetime);

   If &File.SetFileLayout(FileLayout.FLYOUT_CLIENTE) Then

      &RS1 = &File.ReadRowset();

      rem ***** Get current PeopleSoft Session *****;
      &oSession = %Session;
      &oSession.PSMessagesMode = 1;

      While &RS1 <> Null

         rem ***** Get the Component Interface *****;
         &oClicuenCi = &oSession.GetCompIntfc(CompIntfc.CLICUEN_CI);
         If &oClicuenCi = Null Then
            errorHandler();
            Exit;
         End-If;

         rem ***** Set the Component Interface Mode *****;
         &oClicuenCi.InteractiveMode = False;
         &oClicuenCi.GetHistoryItems = True;
         &oClicuenCi.EditHistoryItems = False;

         &RS1.GetRow(1).CLIENTE.CopyFieldsTo(&REC1);

         rem ***** Set Component Interface Get/Create Keys *****;

         rem ***** Execute Create ******;
         If Not &oClicuenCi.Create() Then
            rem ***** Unable to Create Component Interface for the Add keys
provided. *****;
            errorHandler();
            Exit;
         End-If;

         &oClicuenCi.ID = &REC1.ID.Value;
         &oClicuenCi.NOMBRE = &REC1.NOMBRE.Value;
         &oClicuenCi.FECHA_ALTA = &REC1.FECHA_ALTA.Value;

         rem ***** End: Get/Set Component Interface Properties *****;

         rem ***** Execute Save *****;
         If Not &oClicuenCi.Save() Then
            errorHandler();
            Exit;
         End-If;

         &RowsImported = &RowsImported + 1;

         &RS1 = &File.ReadRowset();

      End-While;

   End-If;

   MessageBox(0, "Importacion", 20002, 1, "Message", %Datetime);

   MessageBox(0, "Importacion", 20001, 1, "Message", &RowsImported);

   &File.Close();

   &LogFile.WriteLine("End");
   &LogFile.Close();

End-If;
```

6.12.4.4 Fichero Plano Con Datamover → Tabla

- CLIENTES_IMPORT.dms: Programa que lee de un fichero csv y escribe en una tabla con DataMover.

Fichero a tabla con DataMover

La tabla PS_CLIENTE no deberá contener los registros que queremos importar para que no se produzcan errores de duplicidad de claves. Es aconsejable que esté vacía.

Como el fichero a importar (clientes.dat) tiene que tener un formato específico de PeopleCode es necesario haber realizado una exportación antes de los datos de la tabla PS_CLIENTE.

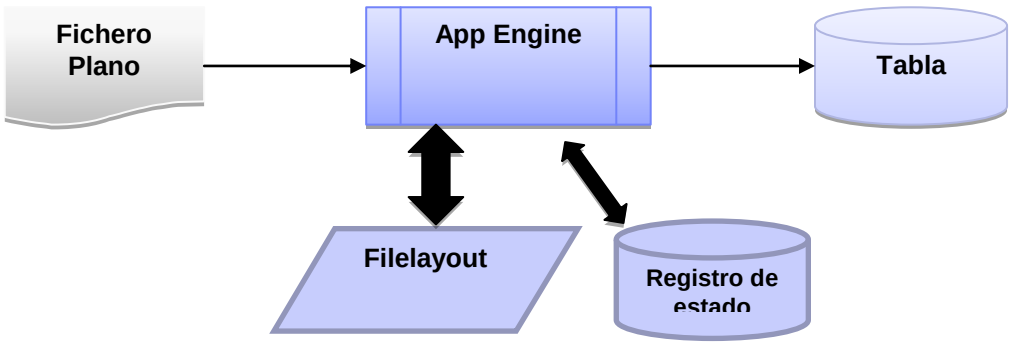
6.12.4.5 Script Data Mover

Con Set Log indicamos la ruta y el fichero donde estará el fichero log y con Set Input indicamos ruta y fichero de donde se leerán los datos que se introducirán en la tabla PS_CLIENTE. Para ejecutar este script necesitamos la aplicación DataMover.

```
SET NO TRACE;
SET LOG C:\TEMP\EXPPROJ.LOG;
SET INPUT C:\TEMP\CLIENTES.DAT;

IMPORT PS_CLIENTE;
```

6.12.5 Programa Re-Arrancable

- APP_FLYIMP_R: Programa que lee de un fichero y escribe en una tabla en modo de re-arranque (Utilizando Filelayout). Con el modo de rearranque si al hacer una importación se produce una interrupción en el proceso y éste queda sin terminar, en la siguiente ejecución de ese proceso podrá terminar la importación de los registros del fichero plano.



Programa rearrancable

6.12.5.1 Definición del Registro de Estado

Este programa utiliza el registro de estado APP_FLY_IMP_AET para guardar datos de cada proceso que sirvan para el re-arranque (en caso de fallo).

El registro de estado está mapeado en una tabla física (PS_ APP_FLY_IMP_AET) y donde el campo PROCESS_INSTANCE es el campo clave.

APP_FLY_IMP_AET (Record)

Record Fields | Record Type

Num	Field Name	Type	Len	Format	Short Name	Long Name
1	PROCESS_INSTANCE	Nbr	10		Instance	Process Instance
2	OPRID	Char	30	Mixed	User	User ID
3	RUN_CNTL_ID	Char	30	Mixed	Run Cntl	Run Control ID
4	FILE_POSITION	Nbr	10			
5	FILE_PATH	Char	70	Mixed		
6	ROWS_PROCESS	Nbr	6			
7	I_ROW_PROCESS	Nbr	6			
8	END_OF_FILE	Char	1	Upper		

Registro de estado

1. **PROCESS_INSTANCE**: Instancia del Proceso.
2. **OPRID**: ID del Usuario.
3. **RUN_CNTL_ID**: Run Control ID.
4. **FILE_POSITION**: Posición de lectura

5. **FILE_PATH**: Ruta del fichero plano. Este campo es parametrizable desde la página *Application Engine Request*.

6. **ROWS_PROCESS**: Número de filas que se procesarán dentro del paso st_ Loop (004). Este campo es parametrizable desde la página *Application Engine Request*.

7. **I_ROW_PROCESS**: Número de fila en el que se interrumpió el proceso dentro de cada grupo de filas que se procesarán en el paso st_Loop (004).

8. **END_OF_FILE**: Este campo sirve para controlar si se ha acabado de leer el fichero plano. Este campo es parametrizable desde la página *Application Engine Request*.

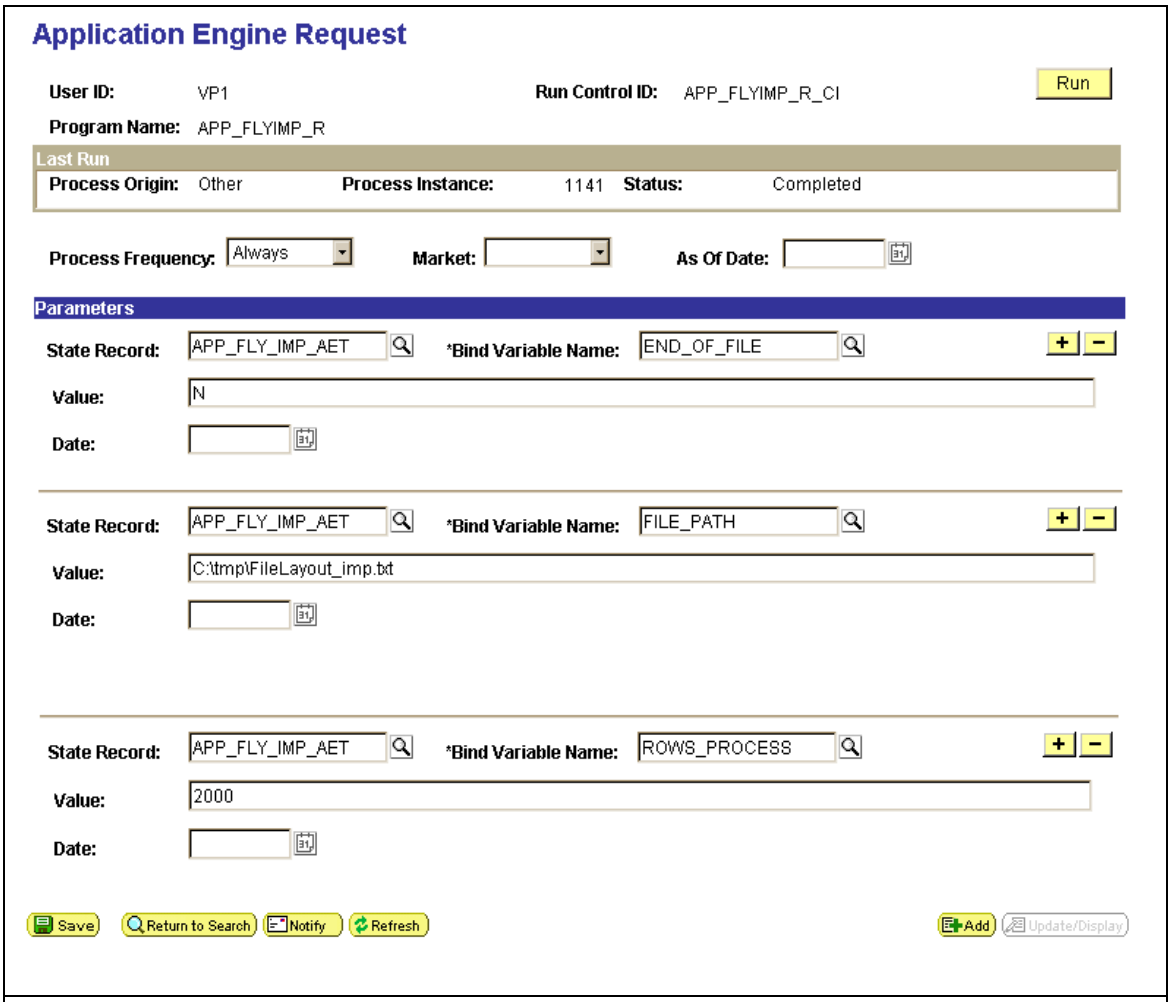
6.12.5.2 Definición de Parámetros en el Programa

Las variables que utilizaremos con la lógica de re-arranque y necesitamos parametrizar en la página ***PeopleTools/Application Engine/Request AE*** del portal son las siguientes:

FILE_PATH: El fichero del cual se leerán los datos. Ej. c:\tmp\FileLayout_imp.txt

ROWS_PROCESS: Número de filas que se procesarán dentro del paso st_ Loop (004). Este campo es parametrizable desde la página *Application Engine Request*.

END_OF_FILE: Este campo sirve para controlar si se ha acabado de leer el fichero plano.



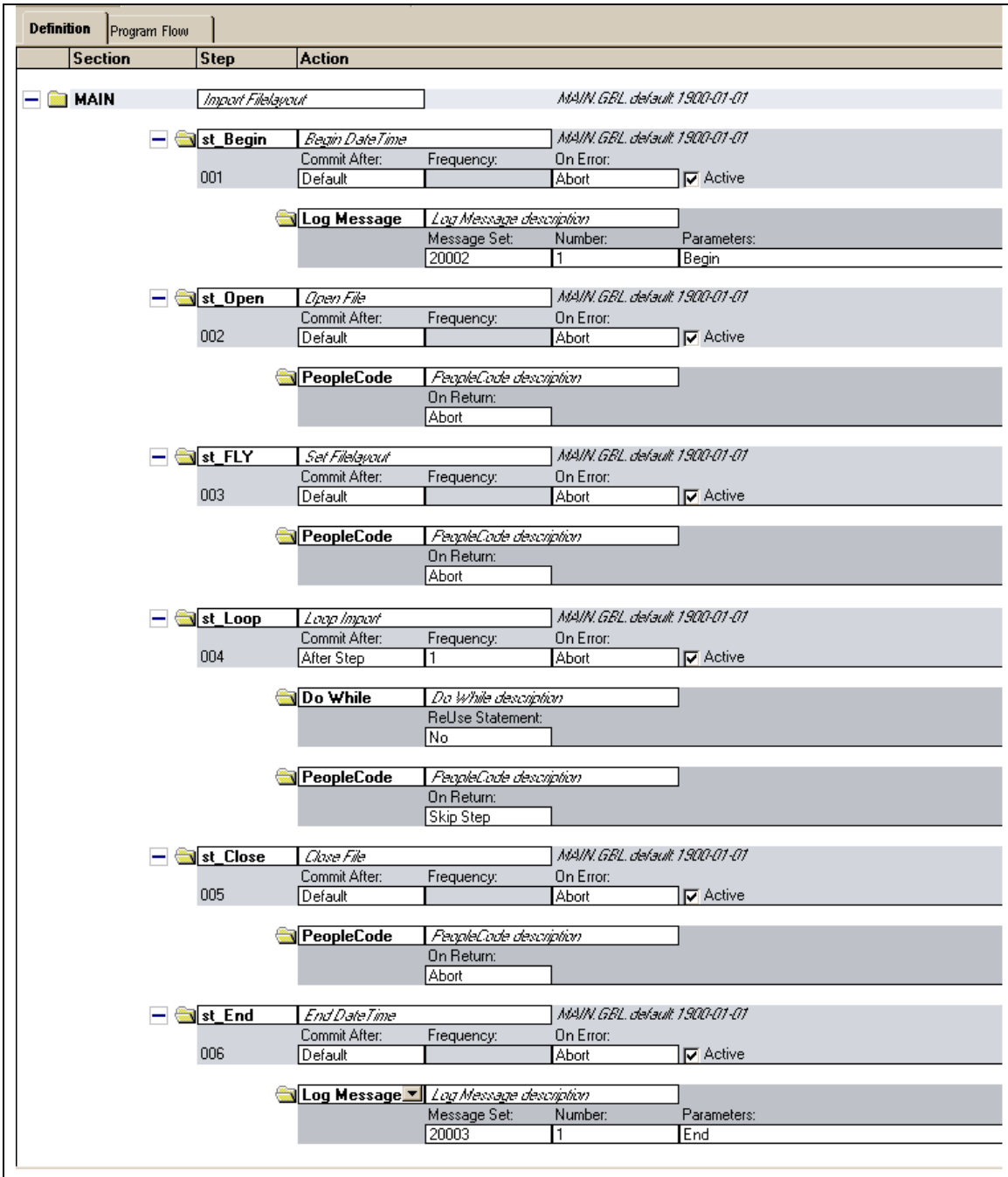
Definición de parámetros

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con los siguientes pasos:

- st_Begin, en este paso se genera un registro en el fichero log. Se utiliza el mensaje 1 del set 20002, previamente definido en el Catálogo de Mensajes. Se introduce el parámetro Begin, para posteriormente localizar la hora de comienzo del programa.
- st_Open, recupera la ruta y el fichero del registro de estado y lo abre en modo update. También crea el registro.
- st_FLY, enlaza el fichero abierto con el Filelayout que utilizaremos para la importación.

- st_Loop, se realiza una acción PeopleCode mientras la sentencia select devuelva datos. La sentencia select para el usuario, la instancia del proceso y para el campo del registro de estado END_OF_FILE siempre devuelve datos, porque este campo lo hemos parametrizado con “N”. Cuando se ejecute la acción PeopleCode y leamos el fin de fichero marcaremos el campo del registro de estado con “Y”, para así salir del bucle y dejar de ejecutar el código PeopleCode. Dentro de este código es donde se van procesando filas en bloques de N registros (según el valor del campo ROW_PROCESS en el registro de estado). Se va haciendo commit al acabar el paso según lo parametrizado en el campo Frequency, en este caso después de cada bloque procesado.
- st_Close, cierra el fichero y mostramos un mensaje con el número de registros importados.
- st_End, en este paso se genera un registro en el fichero log. Se utiliza el mensaje 1 del set 20003, previamente definido en el Catálogo de Mensajes. Se introduce el parámetro End, para posteriormente localizar la hora de fin del programa.



Secciones del programa

2.- CÓDIGO DEL PROGRAMA:

002. Step st_Open: Open File

```
Global number &RowsImported;
Global File &File;
Global Record &REC1;
Global Rowset &RS1;
Local string &FilePath;

MessageBox(0, "Importacion", 20004, 1, "Import", "Begin-" | %Datetime);

&RUN_CNTL_ID = APP_FLY_IMP_AET.RUN_CNTL_ID.Value;
&PROCESS_INSTANCE = APP_FLY_IMP_AET.PROCESS_INSTANCE.Value;
&FilePath = APP_FLY_IMP_AET.FILE_PATH;

&RowsImported = 0;

&File = GetFile(&FilePath, "U", "A", %FilePath_Absolute);
&REC1 = CreateRecord(Record.CLIENTE);
```

003. Step st_FLY: Set Filelayout

```
Global number &RowsImported;
Global File &File;

If &File.IsOpen Then

   If Not &File.SetFileLayout(FileLayout.FLYOUT_CLIENTE) Then

      MessageBox(0, "Importacion", 20004, 1, "Import", "ERROR: Set
Filelayout");
      Exit (1);

   End-If;

Else
   MessageBox(0, "Importacion", 20004, 1, "Import", "ERROR: File Closed");
End-If;
```

004. Step st_Loop: Loop Import

- **Do While**

```
%Select(OPRID)
SELECT OPRID
FROM %Table(APP_FLY_IMP_AET)
WHERE PROCESS_INSTANCE = %ProcessInstance
AND OPRID = %OperatorId
AND RUN_CNTL_ID = 'APP_FLYIMP_R_CI'
AND END_OF_FILE <> 'Y'
```

- **PeopleCode**

```
Global number &RowsImported;
Global File &File;
```

```
Local Record &REC1;
Local Rowset &RS1;
Local SQL &SQL1;

&SQL1 = CreateSQL("%Insert(:1)");
&REC1 = CreateRecord(Record.CLIENTE);

/****** BULKMODE  **********/
&SQL1.BulkMode = True;


If &SQL1.IsOpen Then

   &ROWSPROC = 1;
   &TOTAL_ROWSPROC = APP_FLY_IMP_AET.ROWS_PROCESS.Value;

   &CURPOS = APP_FLY_IMP_AET.FILE_POSITION.Value;
   &File.SetPosition(&CURPOS);

   While &ROWSPROC <= &TOTAL_ROWSPROC

      &RS1 = &File.ReadRowset();

      If &RS1 <> Null Then

         &RS1.GetRow(1).CLIENTE.CopyFieldsTo(&REC1);


         If &RS1.GetRow(1).IsEditError Then

            For &I = 1 To &REC1.FieldCount

               If &REC1.GetField(&I).EditError Then
                  MessageBox(0, "Importacion", 20004, 1, "Message", "ERROR in
field: " | &REC1.GetField(&I).Name);
                  Exit;
               End-If;

            End-For;
         End-If;

         &TODO_OK = &SQL1.Execute(&REC1);

         If Not &TODO_OK Then
            MessageBox(0, "Importacion", 20004, 1, "Message", "ERROR !");

            Evaluate &SQL1.Status
            When = %SQLStatus_OK
               MessageBox(0, "Importacion", 20004, 1, "Message", "It
worked.");
               Exit;
            When = %SQLStatus_NotFound
               MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key
were not found.");
               Exit;
            When = %SQLStatus_Duplicate
               MessageBox(0, "Importacion", 20004, 1, "Message", "The ID key
were already there.");
               Exit;
            End-Evaluate;

         End-If;

         &RowsImported = &RowsImported + 1;

      Else

         APP_FLY_IMP_AET.END_OF_FILE.Value = "Y";
```

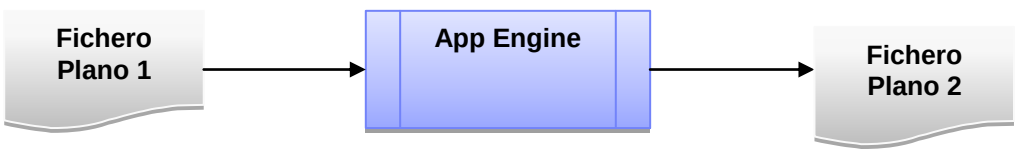
```
         &SQL1.Close();

         MessageBox(0, "Importacion", 20004, 1, "Message", "EOF: " |
APP_FLY_IMP_AET.END_OF_FILE.Value);

         Exit (1);

      End-If;

      &ROWSPROC = &ROWSPROC + 1;

   End-While;

   &CURPOS = &File.GetPosition();
   APP_FLY_IMP_AET.FILE_POSITION.Value = &CURPOS;
   APP_FLY_IMP_AET.I_ROW_PROCESS.Value = &ROWSPROC;
   MessageBox(0, "Importacion", 20004, 1, "Message", "Importadas: " |
&RowsImported);

Else

   MessageBox(0, "Importacion", 20004, 1, "Import", "ERROR: SQL Closed");

End-If;
```

005.Step st_Close: Close File

```
Global number &RowsImported;
Global File &File;

&File.Close();

MessageBox(0, "Importacion", 20001, 1, "Message", &RowsImported);

MessageBox(0, "Importacion", 20004, 1, "Import", "End-" | %Datetime);
```

6.12.6 Descripción de programas: copia de fichero plano

- APP_PLN_PLN: Programa que lee de un fichero y escribe en un fichero. Este programa es re-arrancable en el sentido de que si en algún momento de la ejecución el proceso se interrumpe, aunque el fichero plano creado no contenga todas las filas de la tabla bastaría con volver a ejecutar el programa y reescribiría el contenido del fichero.

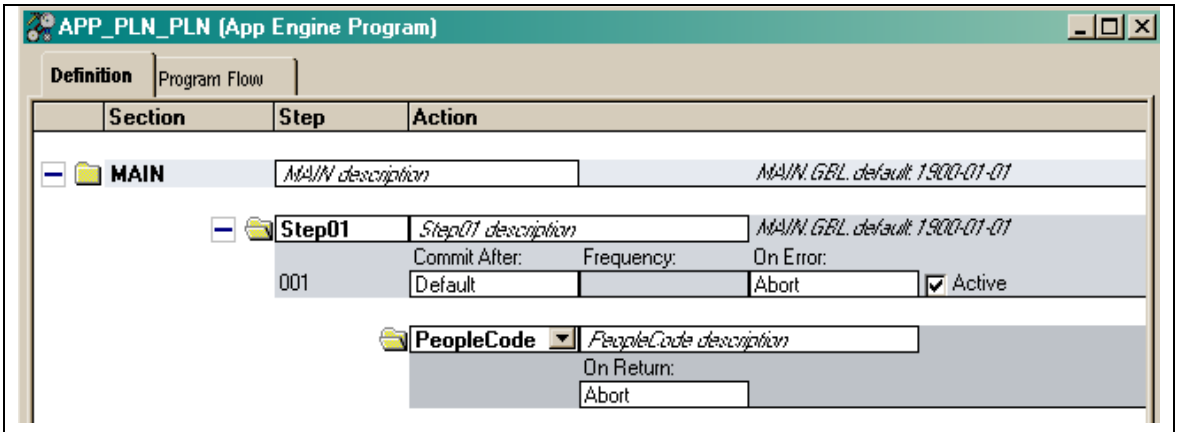


Fich – App Engine -- Fich

El fichero clientes.txt tendrá que contener los registros que queremos copiar. El formateo de los campo se hace por código.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura programa

2.- CÓDIGO DEL PROGRAMA:

Primero creamos los dos ficheros, el origen en modo lectura y el destino en modo escritura. Después vamos leyendo los registros del fichero origen, y por cada fila con la sentencia Split dividimos la fila leída por el separador "," y el resultado se introduce en un array.

El array con los datos de la línea lo vamos leyendo con la sentencia Shift, que va recuperando cada dato y rotando para leer los siguientes. Cuando recuperamos el tercer campo, que es la fecha de alta, lo formateamos como "yyyy-mm-dd". En un campo de tipo String concatenamos los datos leídos separados por "," y escribimos la variable String con la sentencia WriteLine.

Al finalizar cerramos los dos ficheros.

```
Local File &INFILE;
Local File &OUTFILE;

&INFILE = GetFile("C:\tmp\clientes.txt", "R", "A", %FilePath_Absolute);
&OUTFILE = GetFile("C:\tmp\copia_cli.txt", "W", "A", %FilePath_Absolute);

If &INFILE.IsOpen And
      &OUTFILE.IsOpen Then
```

```
   &NUM_MOV = 0;

   While &INFILE.ReadLine(&LINEA)

      &CAMPOS = Split(&LINEA, ",");

      &LEN_ARR = &CAMPOS.Len;

      &ID = "";
      &NOMBRE = "";
      &FECH_ALTA = "";

      For &I = 1 To &LEN_ARR

         &ITEM = &CAMPOS.Shift();

         Evaluate &I
         When 1
            &ID = "ID=" | &ITEM;
         When 2
            &NOMBRE = "NOMBRE=" | Upper(&ITEM);
         When 3
            &DIA = Substring(&ITEM, 1, 2);
            &MES = Substring(&ITEM, 7, 2);
            &ANIO = Substring(&ITEM, 4, 2);
            &FECH_ALTA = "FECH_ALTA=" | &DIA | "-" | &MES | "-" | &ANIO;
         End-Evaluate;


      End-For;

      &NUM_MOV = &NUM_MOV + 1;
      &NEWLINE = &FECH_ALTA | "," | &NOMBRE | "," | &ID | ",COD_MOV=" |
&NUM_MOV;

      &OUTFILE.WriteLine(&NEWLINE);
      &RowsImported = &RowsImported + 1;

   End-While;

   &INFILE.Close();
   &OUTFILE.Close();

End-If
```

6.12.7 Descripción de programas: copia de tabla

- APP_TBL_TBL: Programa que lee de una tabla y escribe en otra tabla de PeopleSoft. Este programa utilizado en modo multi-usuario no es re-arrancable, si en algún momento de la ejecución el proceso se interrumpe podría haber inconsistencia de datos.

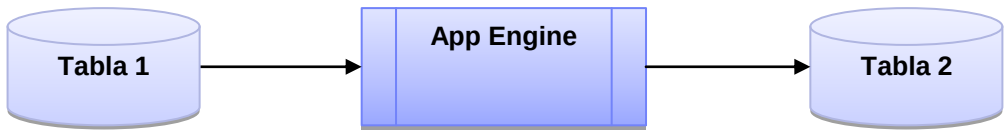
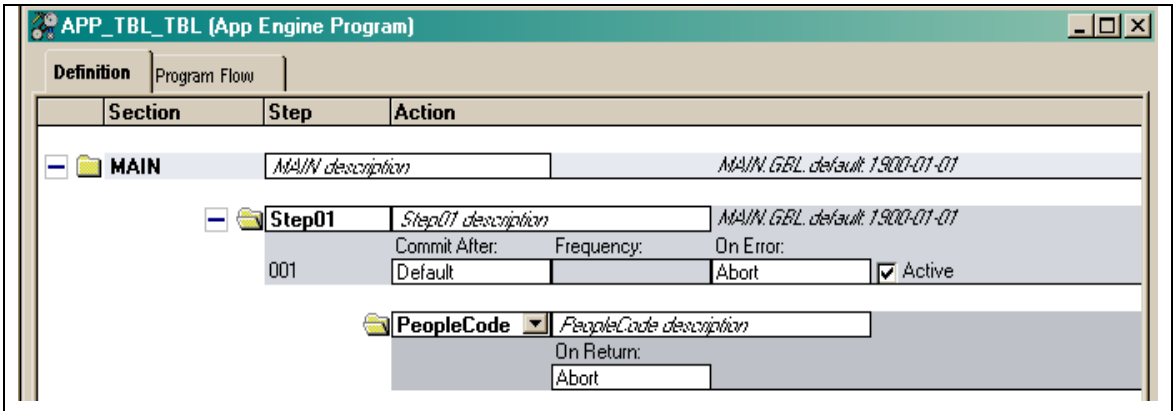


Tabla – App Engine -- Tabla

La tabla PS_CLIENTE tendrá que contener los registros que queremos copiar. El formateo de los campo se hace por código. La tabla PS_CLIENTE_MOV no deberá contener los registros que queremos importar para que no se produzcan errores de duplicidad de claves. Es aconsejable que esté vacía.

1.- ESTRUCTURA DEL PROGRAMA:

El programa tendrá una sola sección (MAIN) con un paso y una acción PeopleCode.



Estructura programa

2.- CÓDIGO DEL PROGRAMA:

Este programa crea dos registros, uno para el origen y otro para el destino, después realiza una sentencia “SELECT * FROM PS_CLIENTE” y por cada fila realiza un insert con los datos del registro leído.

```
Global number &RowsExported;

Local Record &RECIN;
Local Record &RECOUT;

&RECIN = CreateRecord(Record.CLIENTE);
&RECOUT = CreateRecord(Record.CLIENTE_MOV);

&SQL1 = CreateSQL("%Selectall(:1)", &RECIN);
&SQL2 = CreateSQL("%Insert(:1)");
While &SQL1.Fetch(&RECIN)
   &RECOUT.ID.Value = &RECIN.ID.Value;
   &RECOUT.NOMBRE.Value = &RECIN.NOMBRE.Value;
   &RECOUT.FECHA_ALTA.Value = &RECIN.FECHA_ALTA.Value;
   &RECOUT.COD_MOV.Value = 3;
   &SQL2.Execute(&RECOUT);
```

```
   &RowsExported = &RowsExported + 1;

End-While; /* end &SQL1.Fetch */
```

7 CONCLUSIONES Y LÍNEAS DE CONTINUACIÓN

7.1 Conclusiones

La gestión de clientes es fundamental en el éxito de cualquier compañía que ofrezca servicios o productos. En el presente proyecto fin de carrera se ha cumplido los objetivos iniciales del mismo, esto es:

- Dar una explicación y visión general sobre qué es CRM y de los conceptos fundamentales comunes a los sistemas de atención al cliente
- Ver cómo los sistemas que los aplican pueden gestionar e interactuar con los clientes de una manera adecuada.
- Se ha hecho foco fundamentalmente en uno de los CRM más extendidos, PeopleSoft CRM, dando una explicación detallada del mismo.
- Se ha especificado la arquitectura técnica de PeopleSoft viendo cuáles son los componentes que la integran y cómo interactúan entre ellos.
- Se ha presentado un manual de usuario del aplicativo online a modo de introducción para que cualquier usuario pueda entender los conceptos de navegabilidad sobre el portal de PeopleSoft
- Se ha explicado los conceptos claves del lenguaje de programación PeopleCode, un lenguaje potente para el desarrollo de aplicaciones PeopleSoft.
- Se ha detallado en modo manual la programación con dos entidades básicas que engloban al resto de entidades de programación de PeopleSoft y gracias a las cuales se ha podido entender cómo se programa en PeopleSoft: desarrollo de páginas y desarrollo de aplicaciones batch con Application Engine

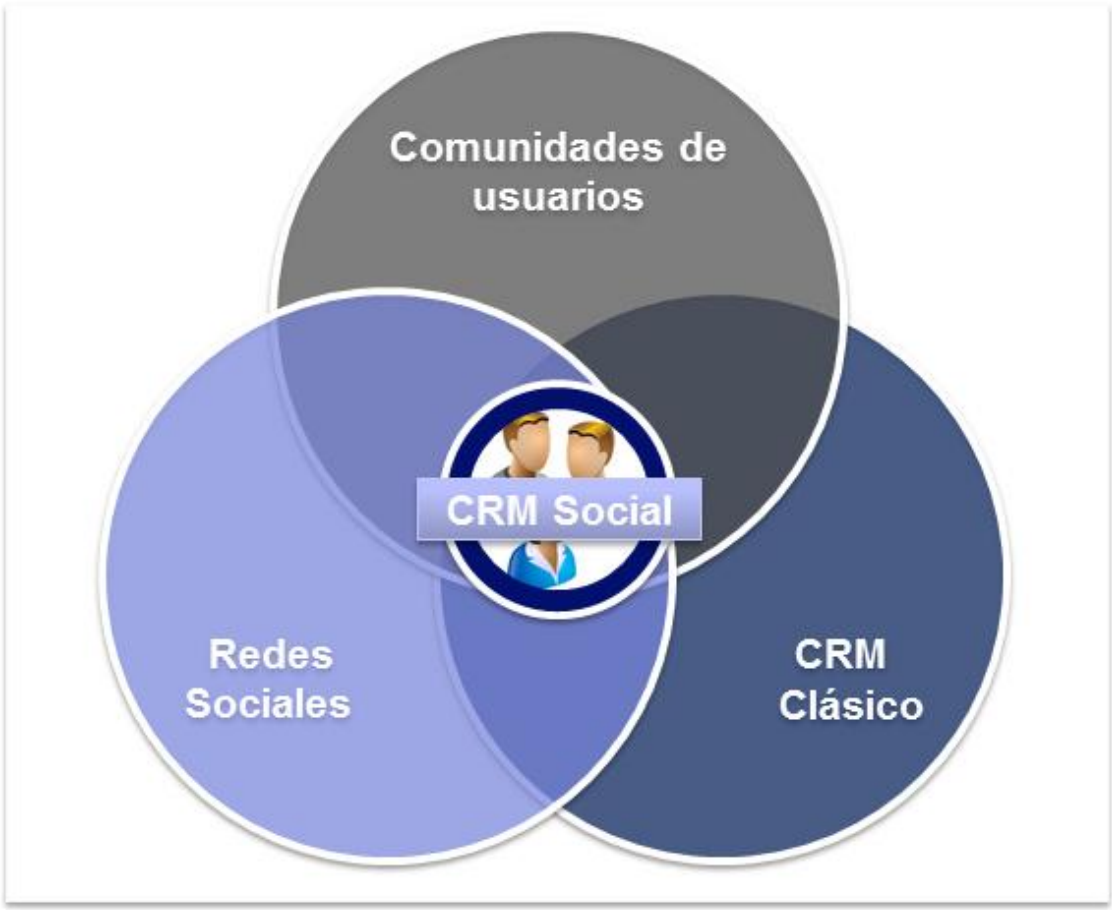
Los sistemas CRM van a seguir siendo claves en las empresas para poder hacer una buena gestión de los clientes. Gracias a estos sistemas se podrá seguir identificando a nuevos clientes, retener y fidelizar a los actuales, dar servicio ante problemas a los mismos y reducir los costes de gestión en la atención al cliente. Esta ha sido la clave de su éxito y lo que les ha

hecho prosperar en los últimos años. Por ese motivo grandes empresas de software como Oracle siguen apostando por el CRM y mantienen esta línea de negocio.

Respecto al caso concreto de PeopleSoft, ¿cómo se vislumbra el futuro? Oracle ha indicado que va a seguir evolucionando el producto a medio y largo plazo y que cada vez estará más integrado con el resto de soluciones de la compañía, fundamentalmente con las nuevas aplicaciones FUSION que son una suite de productos desarrollada por Oracle con productos propios y como fruto de la adquisición de distintas compañías. Esta suite ofrece soluciones CRM, financieras, de Human Capital Management y una amplia variedad de aplicaciones para el ámbito empresarial. PeopleSoft ya tiene una fuerte integración con dicha suite y seguirá la coexistencia de ambas así como de otras soluciones CRM que también ha adquirido Oracle como Siebel CRM.

El respaldo de una gran compañía como Oracle y las grandes inversiones realizadas en los últimos años por empresas en la implantación de herramientas CRM, hace pensar que hay un buen futuro para las aplicaciones CRM y que por tanto será una línea de negocio que se mantendrá activa a lo largo de los próximos años.

En los últimos años ha tomado peso un nuevo concepto relativo al CRM. Podemos ver gran cantidad de artículos y opiniones sobre “Social CRM”. El CRM Social lo que hace es aprovechar las técnicas clásicas de CRM y aplicarlas al ámbito de la redes sociales.



CRM Social

Por todos es conocido el auge que han tenido las redes sociales y comunidades de usuarios en los últimos años. El CRM clásico también ha sabido ver este auge adaptándose y creando el nuevo concepto de CRM Social. La diferencia con el CRM tradicional es que el CRM Social utiliza las redes sociales y comunidades añadiendo la posibilidad de intercambio de información y conversación directa con el cliente a través de estos medios. Las empresas pueden llevar un seguimiento de los contactos, se les da información y soporte y se les premia al producir contenido positivo.

Los seguidores o clientes de una red social habitualmente dan su opinión sobre los productos y servicios que tienen contratados y quieren ser atendidos de forma adecuada. Es aquí donde entra el CRM Social, escuchando las opiniones de los clientes, ya sean positivas o negativas, para adaptar los productos y procesos y así fidelizar u obtener nuevos clientes.

Las técnicas tradicionales de CRM junto a los nuevos conceptos como CRM Social sitúan al cliente como eje central. Por eso, todo indica que a corto plazo como en el futuro se seguirán desarrollando aplicativos que soporten dichas técnicas y que continúen logrando poner al cliente en el centro del negocio.

7.2 Líneas de continuación

Aunque en el proyecto se han cumplido los objetivos marcados al principio del mismo, sí es cierto que al avanzar en su elaboración han ido surgiendo una serie de puntos que merecen una línea de acción y continuación más exhaustiva.

A lo largo de la investigación sobre los sistemas CRM, me he ido encontrado cada vez con más asiduidad con un término que se ha puesto muy en boga en los últimos años y del que hemos hablado en el apartado de conclusiones: el CRM Social. Como primera línea de acción veo necesario realizar una investigación más en profundidad sobre este concepto, los aplicativos que cubren dicha estrategia y cuál es su forma de trabajar para poder aunar el uso de estrategias CRM con las redes sociales.

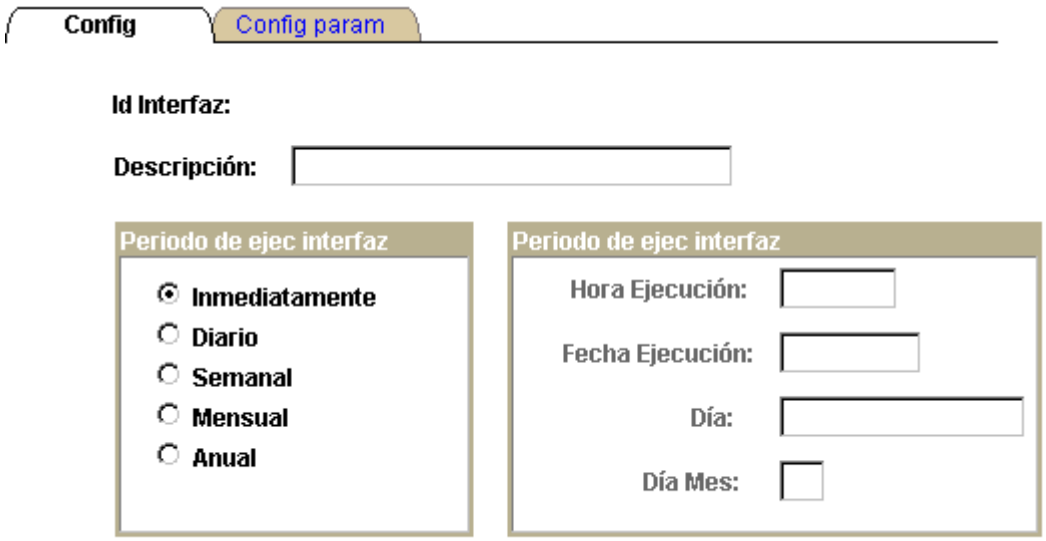
Por este motivo, como primera línea de continuación propongo:

- **Investigación sobre CRM Social,** sistemas que lo implementan y forma de que tienen de trabajar para implementar la estrategia CRM.

Con el trabajo continuado con PeopleSoft CRM en este proyecto también he visto ciertos puntos de mejora en el aplicativo que se podrían cubrir con algunos desarrollos. Los puntos de mejora de desarrollo y líneas de continuación a seguir serían:

- **Implementación de mejora en el planificador de tareas**: uno de los puntos débiles que tiene PeopleSoft es la facilidad de uso del planificar de tareas. No es posible planificar de forma programadas procesos batch con una interfaz sencilla y amigable. PeopleSoft provee el mecanismo de RunControlId, recurrencias de jobs y procesos, que definen la llamada ventana batch, es decir, una fecha, hora y día determinado a la que se ejecuta el proceso, pero configurar dichos procesos no es intuitivo y sencillo ya que requiere de la definición de objetos predeterminados para especificar el tiempo de recurrencia con el que se ha de ejecutar cada proceso. Como línea de continuación

propongo la implementación de una mejora de dicho planificador. Así, habría que desarrollar una serie de ventas y procesos batch que sean capaces de invocar al planificador de PeopleSoft y que permitan una configuración sencilla utilizando las recurrencias y Jobs estándar de PeopleSoft. Ya he empezado a trabajar en un primer esbozo de cómo sería dicho planificador, aunque habría que profundizar en el mismo e implementar toda la lógica que habría por detrás. Incluyo el esbozo de cómo podría ser la pantalla:



Esbozo de pantalla de mejora de planificador de tareas

- **Implementación de interfaz de conversiones**. Otro de los problemas que me he encontrado es que PeopleSoft no tiene de forma estándar una utilidad o conjunto de pantallas que permitan configurar conversiones de datos. Cuando se trabaja con sistemas terceros y tenemos diversos orígenes de datos es muy habitual que se necesiten hacer conversiones y mapeos de datos ya que cada sistema tercero puede manejar una serie de valores que no tiene por qué coincidir con los nativos de PeopleSoft. Por ejemplo, un sistema tercero puede utilizar un tipo de vía con la codificación “C/”, otro “CL” y sin embargo PeopleSoft utiliza de forma nativa “CALLE”. Como siguiente línea de acción propongo la creación de una serie de páginas configurables en el portal de PeopleSoft que permita la introducción de conversiones y

mapeos para el uso de interfaces con terceros, de esta forma se podrá configurar de forma sencilla las conversiones y mapeos y los programas podrían consultar este modelo de tablas en el caso de necesitar conversiones. De esta forma habría que crear un conjunto de páginas agrupadas en Tabs que contengan la siguiente funcionalidad:

- Página de definición de sistema: en la que se indique el sistema tercero que requiere de conversiones con su descripción.
- Página de definición de campos: en la que se indique los campos de PeopleSoft que pueden llevar conversión y si es de entrada o salida.
- Página de mapeos de campo de entrada: campos definidos con mapeos de entrada en los que se configura la fecha efectiva para su utilización, valores del sistema externo, valores de PeopleSoft y estado.
- Página de mapeos de campo de salida; igual que la página anterior pero con los valores de conversión para interfaces de salida.

Además habría que crear una librería con funciones para hacer uso de esta funcionalidad de cara a que pudiera ser invocada por cualquier proceso de forma simple.

- **Mejora en las colas de mensajería de Integration Broker:** PeopleSoft provee de forma nativa Integration Broker como tecnología middleware SOA (Service Oriented Architecture) que facilita el intercambio de mensajes, tanto síncronos como asíncronos, entre sistemas. Dicha tecnología utiliza una serie de colas en la que se recibe la mensajería SOA para ser tratada. Un problema que me he visto con dicha mensajería al revisar la documentación de Integration Broker es que cuando un mensaje cae en error por cualquier causa, la cola deja de procesar los mensajes y va encolando todo lo que llegue para un relanzamiento posterior. Este tipo de procesamiento podría ser útil, siempre y cuando existan alarmas y de forma manual alguien inmediatamente entre a analizar el problema en las colas, elimine el mensaje en error y se siga procesando el resto de la mensajería. Sin embargo, no siempre es posible revisar los errores inmediatamente y por tanto las colas estarían paradas ocasionando el retraso en el

tratamiento del resto de los mensajes. Como línea de continuación propongo la creación un programa Application Engine que esté monitorizando dichas colas ejecutándose de forma concurrente. Cuando exista un error en estas colas el Application Engine levantará una alarma mandando una notificación al administrador del sistema y además historificará el mensaje que ha provocado el error en la cola permitiendo la inmediata ejecución del resto de mensajes.

BIBLIOGRAFÍA

LIBROS:

Título: CRM Fundamentals
Autor: Scott Kostojohn, Mathew Johnson, Brian Paulen
Editorial: Apress
ISBN: 978-1-4302-3590-3

Título: The CRM Handbook: A Business Guide to Customer Relationship Management
Autor: Jull Dyché
Editorial: Addison-Wesley
ISBN: 0-201-73062-6

Título: PeopleSoft Developer's Guide for PeopleTools & PeopleCode
Autor: Doolitle, Judi
Editorial: McGraw Hill Professional
ISBN: 978-0-0716-4357-3

Título: PeopleBooks CRM
Autor: Oracle
ISBN: N/A (documentación del producto)

Título: PeopleTools
Autor: Oracle
ISBN: N/A (documentación del producto)

Título: PeopleSoft Redpapers - Understanding the Business Object Relationship Model for CRM
Autor: Oracle
ISBN: N/A (documentación del producto)

Título: PeopleSoft Redpapers - Extending the Customer Data Model
Autor: Oracle
ISBN: N/A (documentación del producto)

URLs de REFERENCIA:

- Portal Oracle de PeopleSoft: http://docs.oracle.com/cd/E52319_01/infoportal/index.html
- CRM Forum: http://www.crm-forum.com
- CRM español (información y recursos CRM en español): http://www.crmespanol.com/
- PeopleSoft Enterprise PeopleTools Resource Library: http://www.oracle.com/us/products/applications/peoplesoft-enterprise/ptools-resource-library/054014.html
- PeopleSoft Wiki (página con recursos PeopleSoft): http://peoplesoft.wikidot.com/
- Blog en español de un consultor CRM con amplia información: http://www.jesushoyos.com
- Customer Think (artículos sobre CRM): http://customerthink.com
- Inside CRM (artículos sobre CRM): http://www.insidecrm.com/
- CRM Daily (noticias CRM): http://www.crmdaily.com/

PEOPLESOFT EN LAS REDES SOCIALES:

Medio	URL
	**PeopleSoft en YouTube:** https://www.youtube.com/user/PSFTOracle
	**Página de Facebook de PeopleSoft:** https://www.facebook.com/pages/Oracle-PeopleSoft/220476464680933?ref=ts&fref=ts
	**Sigue a PeopleSoft en Twitter:** https://twitter.com/PeopleSoft_Info
	**PeopleSoft Apps Strategy Blog:** https://blogs.oracle.com/peoplesoft/
	**Grupo Linkedin de PeopleSoft:** https://www.linkedin.com/groups/Oracle-PeopleSoft-Development-4530781?home=&gid=4530781&trk=anet_ug_hm